VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřící a řídící techniky

# Rozhraní ZigBee pro modulární telemetrický systém „Cerberos“

# ZigBee interface for „Cerberos“ modular telemetric system

2009 Jaroslav Šustek

VŠB - Technická univerzita Ostrava
Katedra měřicí a řídicí techniky

Fakulta elektrotechniky a informatiky
Akademický rok 2008/2009

# Zadání diplomové práce

Student: **Bc. Jaroslav Šustek**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Rozhraní ZigBee pro modulární telemetrický systém "Cerberos"

ZigBee interface for „Cerberos“ modular telemetric system

Zásady pro vypracování:

1. Seznámení se s komunikačním standardem ZigBee a architekturou FPGA
2. Návrh komunikačního protokolu pro systém „Cerberos“
3. Implementace komunikačního rozhraní a řadiče pro navržený protokol do FPGA
4. Experimentální ověření výsledků

Seznam doporučené odborné literatury:

-PINKER, J., POUPA, M.: Číslicové systémy a jazyk VHDL. Ben - technická literatura, 2006. ISBN 80-7300-198-5.
- PARNELL, K., MEHTA, N.: Programmable Logic Design Quick Start Handbook. Xilinx Inc., 2003.
- ASHENDEN, P.: The Designer's Guide to VHDL. Morgan Kaufmann Publishers, 1998. ISBN 1-55860-270-4.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Vladimír Kašík, Ph.D.**

Datum zadání: 30.11.2008

Datum odevzdání: 07.05.2009

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA

prof. Ing. Vilém Srovnal, CSc.
*vedoucí katedry*

prof. Ing. Ivo Vondrák, CSc.
*děkan fakulty*

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě ..... Podpis: .....

Děkuji vedoucímu své diplomové práce Ing. Vladimíru Kašíkovi, Ph.D. za jeho cenné rady a připomínky, kterými přispěl ke vzniku této diplomové práce.

# Abstrakt

Tato diplomová práce se zaobírá návrhem a realizací rozhraní ZigBee pro modulární telemetrický systém „Cerberos“. Jedná se o vytvoření komunikačního rozhraní mezi sériovou linkou (SCI) a ZigBee modulem pomocí technologie FPGA. V úvodu práce jsou popsány jednotlivé použité technologie - standard ZigBee, hradlová pole typu Spartan-3 a popis nastavení komunikace transceiveru CC2480. V další části je popsán návrh komunikačního protokolu pro systém „Cerberos“ – popis komunikace s jednotlivými komunikačními rámci. V následující hlavní části je celkový, podrobný návrh řešení implementace rozhraní a řadičů v jazyce VHDL. Experimentální testování bylo provedeno ve vývojovém prostředí Xilinx ISE ( komponenta Test Bench) a logickým analyzátorem ChipScope. V konečné části této práce jsou návrhy následné vyšší verze pro „Cerberos 2“.

# Klíčová slova

ZigBee, FPGA, VHDL, hradlová pole Spartan-III, transceiver CC2480, Xilinx ISE, ChipScope, , SPI - sériové komunikační rozhraní, SCI - sériové periferní rozhraní, univerzální asynchronní přijímač, vysílač

# Abstract

This diploma thesis deals with the design and implementation of ZigBee interface for modular telemetric system "Cerberos" . The goal of this task is to create the communication interface between serial line (SCI) and the ZigBee module using the FPGA technology. The thesis introduction describes all used technologies - standard ZigBee, gate arrays Spartan-III type and also a description of setting the communication of transceiver CC2480. Next part of this thesis contains a design of "Cerberos" communication protocol - description of communication with individual communication frames. Following main part of this publication deals with the general, detailed project design - the implementation of interfaces and drivers. The trials has been tested in Xilinx ISE (Test Bench component) design system and with the ChipScope logic analyzer. The last part of the thesis there are some suggestions for subsequent, higher versions of the "Cerberos 2" telemetric system.

# Key words

ZigBee, FPGA, VHDL, field-programmable gate array Spartan-III, transceiver CC2480, Xilinx ISE, ChipScope Serial Communication Interface, Serial Peripheral Interface, Universal asynchronous receiver/transmitter

# Seznam použitých zkratek a výrazů

**ad.** – a další

**apod.** – a podobně

**atd.** – a tak dále

**CLBs -** Configurable Logic Blocks

**CLK -** Clock

**CNT_FIFO** – Counter Fifo

**CSMA/CA -** Carrier Sense Multiple Access with Collision Avoidance

**CTR_RD -** Controller read

**CTR_WR** – Controller write

**DDR -** Double Data Rate

**DPM –** Dual port memory

**FE –** Fifo empty

**FF –** Fifo full

**FIFO -** First In, First Out

**IOBs** - Input/Output Blocks

**JTAG -** Joint Test Action Group

**KO –** klopný obvod

**LUTs -** Look-Up Tables

**M_RDY** - Master ready

**MAC -**

**MX -** Multiplexer

**Obr.** – obrázek

**PHY -**

**RAM –** Random access memory

**SCI -** Serial Communication Interface

**SPI** -

**SRAM** - Static Random Access Memory

**S_RDY -** Slave ready

**tzv.** – takzvaný, tak zvaný

**UAR -** Universal asynchronous receiver

**UAT -** Universal asynchronous transmitter

**USART -** (Addressable universal Synchronous Asynchronous Receiver Transmitter

**VHDL -** Very High Speed Integrated Circuit Hardware Description Language)

# Obsah

# 1. Úvod

Cílem této diplomové práce je realizace komunikačního rozhraní a řadiče (v tomto případě tří řadičů) mezi sériovou linkou (SCI) a ZigBee modulem pomocí technologie FPGA pro telemetrický systém sběru dat „Cerberos“. Vstupem do tohoto zařízení je klasická sériová linka s konektorem RS232 a na výstup je připojen transceiver CC2480 se ZigBee modulem na jednom plošném spoji. Celá práce se tedy týká FPGA technologie propojení těchto zařízení, psána v jazyce VHDL.

V prvních dvou hlavních kapitolách je popis daných zařízení a komunikace s nimi. Do těchto kapitol jsou zahrnuty popisy využívaných periferií, teoretické a hlavně pak praktické informace, získané z tvorby této práce o možnostech návrhu bloků do FPGA.

Třetí kapitola je věnována návrhu protokolů, jak přijímaného tak vysílaného ze zařízení. Tyto protokoly, datové rámce procházející zařízením, nejsou nijak měněny, vstupní je „ zabalen“ do výstupního, pro co nejmenší poškození informace.

Další kapitoly se zaměřují na jednotlivé realizované bloky do FPGA, napsány v jazyce VHDL. Jsou to bloky příjmů dat bitů ze sériové linky (blok UAR), jejich deserializace a následné uložení do paměti RAM s organizací FIFO. FIFO jakožto paměť fronty, je realizována jako paměť dvouportová (DPM), tzn. že můžeme data najednou zapisovat i číst. Data jsou pak serializována a vysílána pomocí bloku SCI vysílače (blok AUT) do transceiveru CC2480. O část zápisu se stará řadič příjmu dat (blok CTR_WR) a o nastavení transceiveru a vysílání dat řadič vysílání (blok CTR_RD). Tyto řídící bloky jsou realizovány stavovými automaty.

Celý systém bloků a jejich částí je synchronizován s hlavními hodinami o frekvenci 50 MHz, z důvodu minimalizace hazardů. Z těchto hlavních hodin jsou dále odvozeny frekvence příjmů, vnitřního zpracování a vysílání dat.

Výsledky této práce jsou vyhodnocovány s využitím testovací komponenty programu Xilinx ISE ( část kódů) a logickým analyzátorem ChipScope (část hardwarová).

Tato diplomová práce bude v budoucnu rozšířena o další součásti (přenos různých dat) pro „Cerberos 2“. Návrh, jak toto zařízení rozšířit, je nastíněno v poslední kapitole.

# 2. Popis problému, specifikace cíle

Vytvoření komunikačního rozhraní mezi sériovou linkou (SCI) a ZigBee modulem pomocí FPGA.

- Seznámení se s komunikačním standardem ZigBee a architekturou FPGA.
- Návrh komunikačního protokolu pro systém „Cerberos“.
- Implementace komunikačního rozhraní a řadiče pro navržený protokol do FPGA.
- Experimentální ověření výsledků.

# 3. Rozbor problematiky

## 3.1. Bezdrátový komunikační standard ZigBee

Komunikační technologie ZigBee je jednoduchý nízkorychlostní standard bezdrátové komunikace, který umožňuje vzájemnou komunikaci mnoha zařízení na vzdálenost stovek metrů.

Díky nízkým nárokům na hardware a nízké spotřebě je zaměřený především na oblasti automatizace a řídicí techniky, např. řízení budov, dálkové ovládání, monitorování a diagnostika zařízení, vzdálené čtení měřených hodnot bateriově napájených senzorů., počítačové periferie nebo spotřební elektronika.

V současné době patří mezi nové, specifikace byla vydána v roce 2004, perspektivní komunikační technologie, které se snaží vyplnit mezeru mezi rozšířenými technologiemi WiFi a Bluetooth. Zde je totiž mezera v podobě velké skupiny aplikací, pro které nejsou Bluetooth ani WiFi, příp. Irda, ideálním řešením, i když se dají použít.

### 3.1.1. Specifikace rádiové části standardu

Standard ZigBee je založen na využití fyzické a linkové vrstvy podle mezinárodního standardu IEEE 802.15.4 - patří do skupiny bezdrátových sítí PAN (Personal Area Networks). Bylo pro něj definováno několik rádiových pásem, aby byl akceptovatelný v různých zemích s odlišnými předpisy a kritérii. Základním problémem při definici rádiových pásem jsou především rozdíly v organizaci rádiových pásem v Americe a na evropském kontinentě. Aby se mohl standard uplatnit v obou těchto lokalitách, jsou pro něj definována tři rádiová pásma :

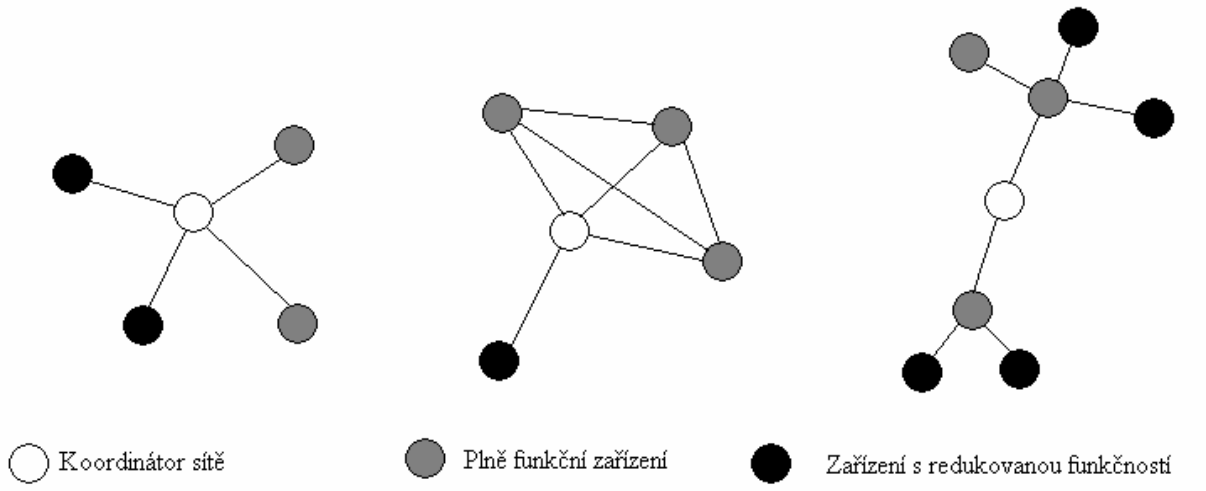
- globální použití: pásmo ISM 2,4 GHz s 16 kanály a přenosovou rychlostí 250 kb/s;
- Amerika a Austrálie: pásmo 915 MHz s 10 kanály a přenosovou rychlostí 40 kb/s;
- Evropa: pásmo 868 MHz s jedním kanálem a přenosovou rychlostí 20 kb/s.

Technologie je založena na implementaci přístupové metody CSMA/CA k fyzickému médiu, což znamená, že vlastní rádiová část standardu IEEE 802.15.4 této metody využívá na úrovni fyzické a linkové vrstvy komunikačního modelu. Vlastní standard IEEE 802.15.4 definuje komplexní komunikační protokol, který je založen na přenosu datových rámců. Jsou definovány čtyři typy komunikačních rámců využívané buď pro přenos užitečných datových informací, nebo k režijním účelům souvisejícím se sestavením, správou a řízením sítě: [2]

- Data Frame – rámec s délkou užitečných dat 104 bajtů slouží pro přenos užitečné informace pro všechny datové přenosy v kontextu standardu;
- Acknowledgement Frame – rámec sloužící pro přenos potvrzovací informace; je využitelný pouze na úrovni MAC pro potvrzovanou komunikaci a je vysílán v takzvaném „mrtvém čase“ ihned po přenosu paketu;
- MAC Command Frame – rámec slouží k centralizovanému konfigurování, nastavení a řízení klientských zařízení v síti ZigBee;
- Beacon Frame – rámec slouží k synchronizaci zařízení v síti a je využíván hlavně při konfiguraci sítě v módu beacon enable, v němž umožňuje uvádění klientských zařízení do spánkových režimů s extrémně sníženou spotřebou.

### 3.1.2. Topologie sítě ZigBee

Standard IEEE 802.15.4 využívá pro adresaci jednotlivých zařízení binární adresovací kódy, které mohou být buď dlouhé (64 bitů), či zkrácené (16 bitů). Lokální adresa zkráceného adresovacího kódu umožňuje v jedné síti adresovat maximálně 65 535 zařízení. Každá sestavená síť je dále identifikována 16 bitovým identifikátorem PAN ID, který slouží pro rozlišení překrývajících se sítí v případě, že v jednom prostoru dochází k vytvoření a sestavení více sítí standardu IEEE 802.15.4. Každou síť s unikátním PAN ID zakládá a spravuje koordinátor (centrální stanice), přičemž ostatní stanice pracují v módu koncové stanice. Každá koncová stanice může být konfigurována pro funkci směrovače nebo koncového zařízení. Podle funkčnosti se zařízení dělí na plně funkční zařízení (FFD), která mohou zastávat funkci koordinátora nebo směrovače, a na redukovaná zařízení RFD, která mohou fungovat pouze jako koncová zařízení.




Obr. 2.1. Topologie sítě ZigBee: typu hvězda (a), síť (b) a strom (c)

Standard ZigBee založený na fyzické a linkové vrstvě IEEE 802.15.4 definuje tři typy síťové topologie (obr. 2.1. ). Základní je topologie typu hvězda (star topology), v níž je vždy definováno jedno zařízení, které přebírá funkci koordinátora sítě, a ostatní zařízení působí ve funkci koncových zařízení. V topologii typu strom (tree topology) slouží jedno zařízení jako koordinátor a ostatní jako koncová zařízení. Na rozdíl od topologie hvězda však nemusí všechna zařízení komunikovat přímo s koordinátorem, ale mohou využít jiné koncové zařízení v konfiguraci FFD ve funkci směrovače jako prostředníka. Díky tomu umožňuje uvedená konfigurace zvětšit vzdálenosti mezi koncovým zařízením a koordinátorem. Poslední definovanou topologií je topologie typu síť (mesh topology), která

kombinuje vlastnosti topologií strom a hvězda (tzv. hybridní topologii strom a hvězda). Síťová topologie přináší největší funkčnost, protože umožňuje sestavit síť libovolným způsobem.

### 3.1.3. Synchronizace zařízení ZigBee

Synchronizace jednotlivých zařízení v síti ZigBee, potažmo koncových zařízení s koordinátorem sítě je realizována na základě tzv. rámce beacon. Synchronizační autoritou je zde koordinátor sítě, který v daných okamžicích vysílá synchronizační sekvence, neboli beacon. Sekvence přijímají ostatní zařízení a synchronizují se podle nich s vysílací stranou, tedy s koordinátorem. Tento postup umožňuje koncová zařízení na dlouhou, předem definovanou, dobu „uspat“ a značně tak snížit jejich spotřebu. [2]

## 3.2. FPGA

Programovatelná hradlová pole, všeobecně známá pod zkratkou FPGA, představují jeden z nejvýznamnějších směrů vývoje integrovaných obvodů s velmi velkou hustotou integrace (VLSI). Programovatelná hradlová pole jsou obvody s pravidelnou strukturou logických buněk schopných realizovat jednoduché logické funkce. Volitelným propojením těchto buněk lze dosáhnout rozsáhlých komplexních funkcí, k jejichž realizaci by jinak bylo nutné použít mnoho různých obvodů. [3]

Pro řešení diplomové práce je použit obvod typu FPGA firmy Xilinx a to programovatelná hradlová pole typu Spartan-3.

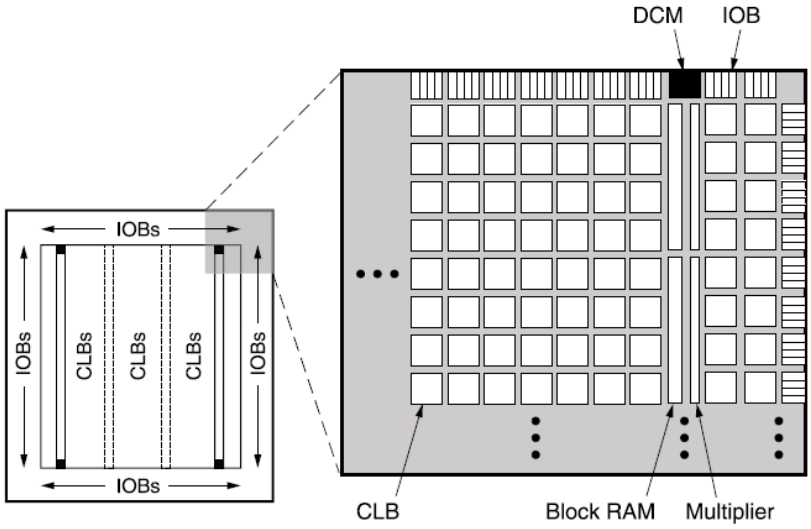
### 3.2.1. Architektura Spartan-3

Základní architektura rodin Spartan obsahuje matici konfigurovatelných logických bloků (CLB), množství lokálních a globálních propojovacích prostředků, vstupně/výstupní bloky (IOB), programovatelné vstupně/výstupní oddělovače a statickou paměť konfigurace.

Každý blok CLB obsahuje tzv. look-up tables (LUT), multiplexory, registry a signálové cesty pro řídící signály. LUT si lze představit jako konfigurovatelnou paměť ROM se čtyřbitovou adresovou a jednobitovou datovou sběrnicí.

Obvody FPGA Spartan jsou implementovány v pravidelné programovatelné architektuře konfigurovatelných logických bloků (Configurable Logic Blocks - CLBs), které jsou vzájemně propojeny rozsáhlou hierarchickou strukturou propojovacích vodičů ( Routing Chanels). Vnější obvody programovatelné struktury bloků CLB tvoří vstupně/výstupní bloky ( Input/Output Blocks – IOBs) a DCM (Digital Clock Manager) poskytující plně digitální řešení distribuce zpoždění

hodinového signálu. Pro ukládaní dat ve formátu 18 Kb dual-port blocks je zde bloková paměť RAM.[4]




Obr. 2.2. Uspořádaní základních bloků v architektury obvodu Spartan 3

### 3.2.2. Popis logiky

#### 3.2.2.1. IOBs (Vstupně výstupní bloky)

IOB poskytuje programovatelné obousměrné rozhranní mezi I/O vývodem a interní logikou FPGA. Struktura IOBs obsahuje 3 základní signálové cesty: vstupní, výstupní a 3-stavovou. [4]

- Vstupní cesta přenáší data z vývodu do vnitřní logiky přes volitelné programovatelné zpoždění na linku I. Další možné výstupy jsou IQ1 a IQ2 přes klopné obvody.
- Výstupní cesta z vnitřní logiky vede přes linky O1 a O2 na výstupní vývod.
- 3-stavový výstup je řízen linkami T1 a T2.

#### 3.2.2.2. CLB (Matice konfigurovatelných logických bloků)

Většina logických funkcí je v FPGA implementována právě v obvodech CLB. Základním stavebním prvkem konfigurovatelného logického bloku (CLB) je logická buňka (Logic Cell - LC). Struktura CLB obsahuje[4]:

- 4 x SLICE (menší logické elementy).
- 2 x nezávislé „carry“ řetězce pro konstrukci rychlých sčítaček.
- Rychlé připojení k sousedním členům.

- Aktuální zapojení dle konfigurace.
- Podporu SelectRAM , 16b RAM nebo 16 bitový shift registr.

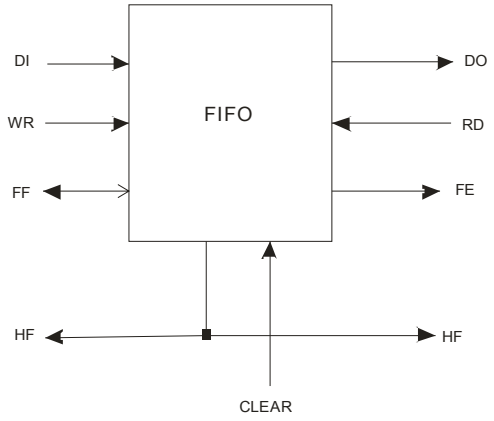
#### 3.2.2.3. Bloková RAM

Obvody SPARTAN-3 obsahují blokovou RAM uspořádanou po blocích velikosti 18 Kbit. Pro uložení velkého množství dat je efektivnější použít blokovou RAM než distribuovanou RAM.

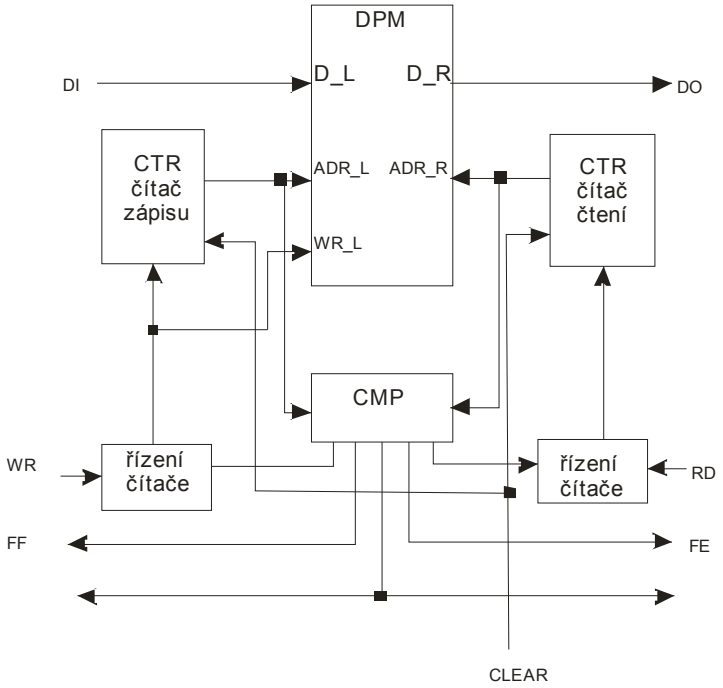
Distribuovaná paměti je realizována z tabulek LUT, jsou vhodné pro nevelké paměťové subsystémy, které nepotřebují ke své funkci tolik kombinační logiky, jako jsou například posuvné registry nebo jedno- či dvoubránové paměti. Operace čtení a zápisu do paměti jsou přitom podstatně rychlejší, než při použití paměti v samostatném pouzdře.

Bloková paměť RAM je výhodná pro realizaci rozsáhlejších paměťových subsystémů na čipu FPGA. Na rozdíl od dříve uvedeného způsobu realizace tento paměťový subsystém téměř nespotřebovává univerzálně použitelné logické prvky z logických buněk, nepočítáme-li nevelké struktury, které jsou v některých aplikacích potřebné pro jeho řízení. Kapacita paměti obvodu SPARTAN-3 je 18432 bitů bez parity nebo 16384 bitů s paritou. Paměť také umožňuje dvouportový přístup – může být konfigurována jako Single-Port nebo Dual-Port RAM. Paměť je na čipu rozmístěna po blocích tvořících sloupce. Vhodná pro uložení většího objemu dat, paměť instrukcí pro procesor, konstrukce FIFO pamětí, FSM obvodů nebo také jako vyrovnávací paměťový člen, apod.[4][1]

Pamět FIFO:




Obr. 2.3. Vstupní a výstupními signály paměti FIFO




Obr. 2.4 Konstrukce paměti FIFO se dvěma ukazateli adres

### 3.2.3. Hlavní rysy architektury SPARTAN-3:

- nová 90 nm technologie výroby
- až 5 miliónů systémových hradel nebo 74880 logických buněk
- hodinový signál až 326 MHz
- až 784 I/O vývodů s 622 Mb/s přenosovou rychlostí
- podpora až 17 logických standardů
- Double Data Rate (DDR) podpora - přenos dat na obou koncích hodinového signálu
- integrovaná struktura pro konstrukci rychlých sčítaček
- 4 až 104 integrovaných násobiček 18 x 18
- až 1872 Kb blokové RAM
- až 520 Kb distribuované RAM
- až 4 bloky Digital Clock Manager (DCM) umožňující frekvenční syntézu
- 8 globálních linek pro rozvod hodinových signálů
- plná podpora vývojového prostředí Xilinx ISE

[5]

### 3.2.4. Popis Spartan-3 Starter Kit Board

Společnost Xilinx dodává vývojové desky s osazeným FPGA a několika základními periferiemi a rozhraními. Příkladem takovéto vývojové platformy je Xilinx Spartan-3 Starter Kit Board, využité při tvorbě této práce. Jedná se o vývojovou desku s FPGA obvodem generace Spartan-3 a základními periferiemi - RAM, PlatformFlash, JTAG rozhraní, RS232, PS/2, VGA, ap.

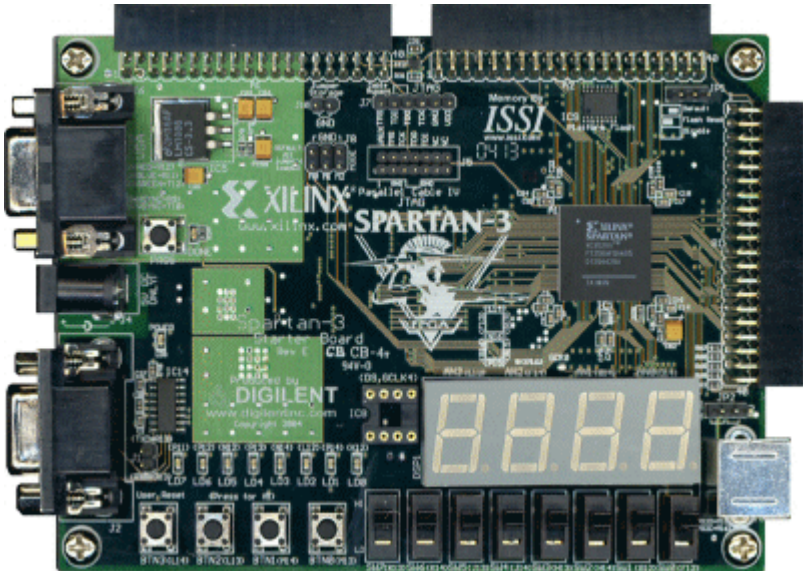
Vývojová deska Spartan-3 je levná platforma pro návrh a testování FPGA designu. Obsahuje základní rozhraní potřebné pro interakci s běžnými periferiemi či přímý vstup uživatele (např. tlačítka).

Hlavní součástí vývojové desky je Xilinx Spartan-3 XC3S200 FPGA. Obsahuje 200 000 ekvivalentních hradel, z dalších zajímavých vlastností lze zmínit integrovanou rychlou paměť BlockRAM.

**Další součásti vývojové desky:**

- Xilinx XCF02S Platform Flash – slouží s trvalému uložení konfiguračního PROM souboru pro FPGA, obvod pracuje hned po zapojení - potřebnou konfiguraci má uloženou v této trvalé paměti
- 1MB SRAM – 2 moduly 256Kx16 SRAM s latencí 10 ns, společná adresní sběrnice
- 3-bitový VGA konektor – umožňuje zobrazovat 8 barev
- 9-pinový RS-232 konektor – viz dále
- PS/2 konektor – možnost připojení myši nebo klávesnice
- 4 sedmisegmentové displeje
- 8 LED diod, 4 tlačítka, 8 přepínačů – viz dále
- 50 MHz krystal – zdroj hodinového signálu
- tlačítko, kterým lze vynutit rekonfiguraci FPGA, standardně se tak děje při připojení napětí
- 3 40-pinové rozšiřující porty – možnost připojení dalších periferních zařízení
- JTAG rozhraní

[5]




Obr. 2.5. Vývojová deska s obvodem FPGA řady Spartan3

### 3.2.5. RS-232

Vývojová deska je vybavena konektorem RS-232 DB9, ke kterému můžeme připojit prodlužovací kabel a propojit desku s počítačem a převodníkem napěťových úrovní MAX232. Umožňuje tudíž bezproblémové využívání sériové linky. Díky zapojení konektoru DB9 na desce máme k dispozici pouze vývody RX a TX. Lze tedy realizovat třížilové spojení s kabelem RX-TX-zem. Na vývodu RX jsou data přijímána, na vývodu TX odesílána. Jedná se o vzájemnou sériovou komunikaci, tzn. jednotlivé datové bity jsou vysílány po jediném vodiči postupně za sebou.

### 3.2.6. Tlačítka, přepínače, LED diody

Tlačítka obsažená na desce reagují na stisk přivedením napětí logické ‘1’. Přepínače lze přepnout trvale do jedné logické polohy. LED diody jsou připojeny přes odpory a svítí při logické ‘1’. Na připojení těchto periferií je použito 4 + 8 + 8 = 20 vývodů.

## 3.3. Metody návrhu logických systémů pro obvody FPGA

### 3.3.1. Návrh v podobě schématu, VHDL

Návrh v podobě zakreslení obvodového schématu logického systému není při tvorbě této práce využit a nebude tedy dále popisován.

Logický systém pro programovatelné hradlové pole je popsán v jazyku typu HDL (Hardware

Description Language). Existují dva silné standardy nabízející návrhářům téměř shodné možnosti a to, u nás velmi rozšířený, jazyk VHDL a jazyk Verilog, oblíbený zejména na americkém kontinentě.[6]

Jazyk VHDL je standardizován, tudíž je snadno přenositelný a téměř nezávislý na cílové platformě. Je možné snadno provést úpravy i v obyčejném textovém editoru. Přesto si schéma zachovává jednu výhodu, a tou je přehlednost vycházející z grafického vyjádření informace. Vhodným řešením může být kombinovaný návrh (mixed design) s hierarchickou strukturou dokumentů, na jejímž vrcholu je právě schéma s hlavními bloky logického systému popsanými jazykem typu HDL nebo opět schématem.

### 3.3.2. Stavový diagram jako popis logického obvodu

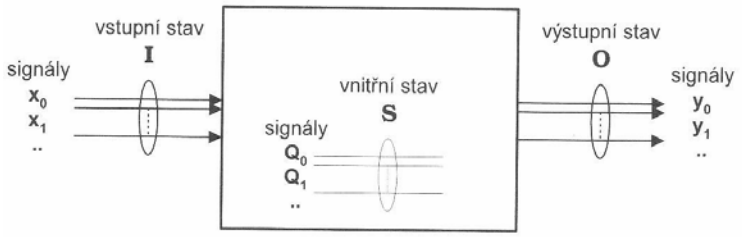
K návrhu konečných automatů přehlednou grafickou formou popisu lze s výhodou použít stavové diagramy. Stavový diagram se v číslicových systémech realizuje vždy jako sekvenční logický obvod, který se může nacházet v jednom z množiny stavů. Tato množina je konečná (odtud název) a přechod mezi jednotlivými stavy závisí na logických hodnotách vstupů v okamžiku aktivní hrany řídicího (taktovacího) signálu. Stavový diagram lze většinou přeložit do některého z jazyků typu HDL. Použití stavového diagramu je výhodné při návrhu řadičů, tedy řídicích jednotek logických systémů a subsystémů.[6]

#### 3.3.2.1. Sekvenční obvody – konečný automat

Základní vlastností sekvenčních obvodů, kterou se liší od obvodů kombinačních je to, že stejné hodnoty vstupních proměnných přivedené na vstup obvodu, nevyvolávají vždy stejnou odezvu na výstupu obvodu. Výstupní signály sekvenčního obvodu jsou tudíž závislé nejen na signálech vstupních v daném okamžiku, ale též na vstupních signálech předcházejících. Jinými slovy - sekvenční obvod má vnitřní paměť.

Abstraktním modelem sekvenčního obvodu je konečný automat - má konečný počet vstupních stavů, výstupních stavů a vnitřních stavů. Jako stav je rozuměna kombinace hodnot signálu. V logických obvodech pracujeme se signály dvouhodnotovými (0 a 1**),** kterých je vždy konečný počet. Proto i počet možných kombinací hodnot těchto signálů je konečný. Konečný automat nemá žádnou další paměť kromě informace o aktuálním stavu.

Obr. 2.6. ukazuje automat se vstupními signály $x_i$, výstupními signály $y_i$, a vnitřními signály $Q_i$,. Vstupní stav je označen jako I, výstupní jako O a vnitřní jako S.




Obr. 2.6. Signály a stavy v sekvenčním obvodu

Závislost následujícího vnitřního stavu na současném vnitřním stavu a vstupním stavu vyjadřuje přechodová funkce:

$$S^{t+1} = f(S^t, I^t) \tag{1}$$

kde symbol $S^{t+1}$ znamená vnitřní stav následující, $S^t$ vnitřní stav současný, $I^t$ vstupní stav současný.

Výstupní stav konečného automatu může být generován několika způsoby. Každý z nich je definován příslušnou výstupní funkcí. V nejobecnějším případě je výstupní stav funkcí vstupního stavu a vnitřního stavu:

$$O^t = g(S^t, I^t) \tag{2}$$

Shodné indexy u všech proměnných znamenají, že výstupní stav je dán vnitřním stavem a vstupním stavem ve stejném okamžiku, jedná se o kombinační funkci. Automat s takto definovanou výstupní funkcí se nazývá **Mealyho automat.**

Další možnost generace výstupního stavu je definována výstupní funkcí závislou jen na vnitřním stavu:

$$O^t = g(S^t) \tag{3}$$

Automat s takto definovanou výstupní funkcí se nazývá **Moorův automat**. Závislost výstupu na minulosti, je zde tvořena prostřednictvím vnitřního stavu, který je závislý na minulých vstupních stavech.

Zvláštním případem je **autonomní automat**, jehož následující vnitřní stav nezávisí na vstupním stavu. Systém se zcela obejde bez vstupních signálů. Využívají se např. u čítačů.

$$S^{t+1} = f(S^t) \tag{4}$$

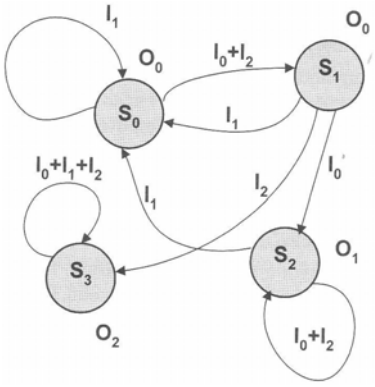
[1]

### 3.3.2.2. Popis pomocí grafu

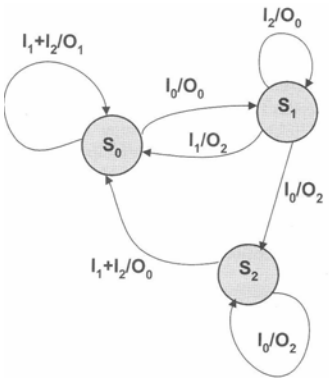
Popsat konečný automat, lze několika způsoby :

- grafické prostředky ( grafy, vývojové diagramy )
- soustavou rovnic
- tabulkami
- v některém programovacím jazyku

Nejpřehlednější je orientovaný graf, který je popsán jazykem typu HDL nebo zakreslením obvodového schématu logického systému. Orientovaný graf vyjadřuje přehledně přechodovou i výstupní funkci. Takovéto grafy používané pro popis stavových automatů se nazývají stavové diagramy. Z uzlu $S_i$ do uzlu $S_j$ vede hrana označená $I_k$, jestliže stav $S_i$ při vstupu $I_k$ přechází do stavu $S_j$. Výstupy u Mealyho automatu se značí u jednotlivých hran grafu (přechodů), neboť výstup závisí na stavu a vstupu. Vstupní stav odpovídá příslušné hraně. Naopak u Moorova automatu se výstupy značí u uzlu grafu (stavu), neboť výstup závisí jen na stavu. Obr 2.7. ilustruje příklad Moorova automatu, obr. 2.8. příklad Mealyho automatu.[1]




Obr. 2.7. Graf Moorův automatu



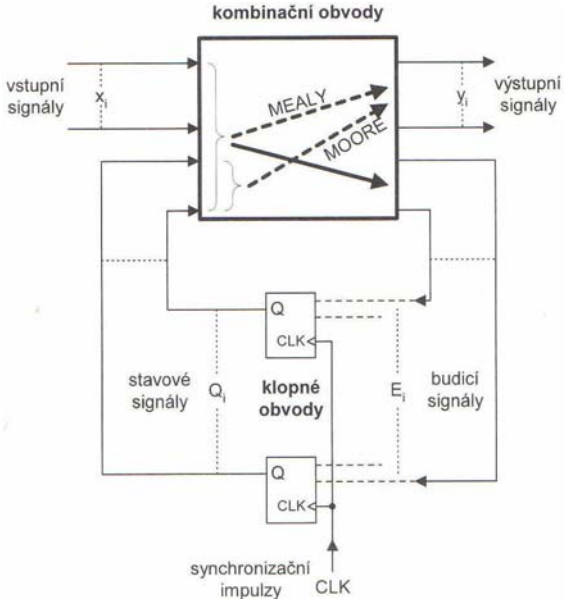

Obr. 2.8. Graf Mealyho automatu

### 3.3.2.3. Obvodová realizace konečného automatu

Obr. 2.9. ukazuje návrh synchronního sekvenčního obvodu – stavového automatu. Vnitřní stav je dán kombinací hodnot signálů $Q_i$ , získaných z klopných obvodů. Pokud známe typ klopných obvodů ( D, T, JK,.., vždy synchronních), je možné ke každému přechodu ze současného do následujícího stavu určit potřebné hodnoty budících signálů klopných obvodů $E_i$. Jelikož následující stav je určen

současným stavem a vstupem v čase t, bude platit: [1]

$$E_i = e_i(Q_0, Q_1, .., x_0, x_1, ...) \tag{5}$$

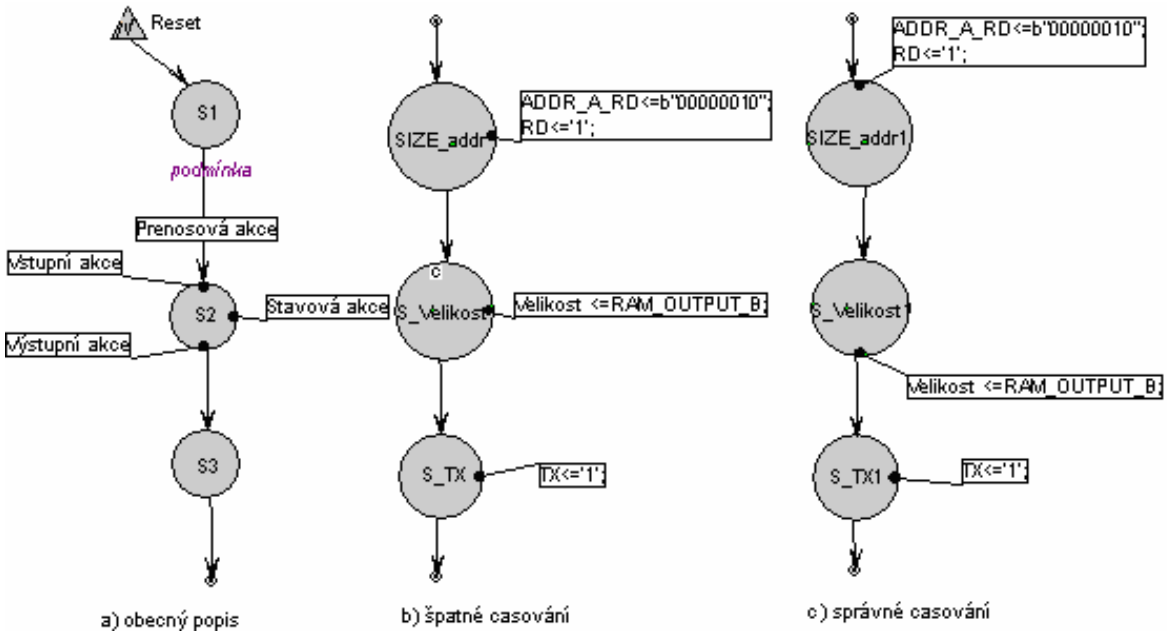



Obr. 2.9. Cesty signálů v synchronním sekvenčním obvodu

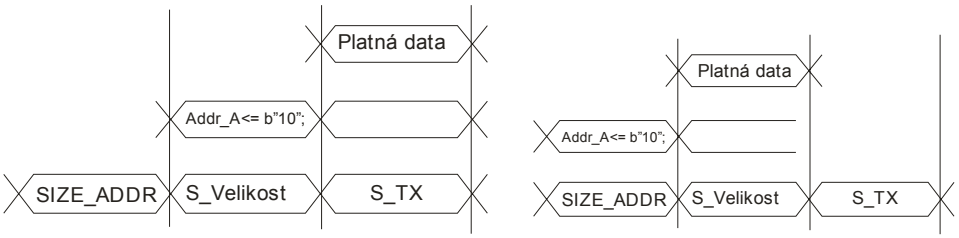
### 3.3.2.4. Časování ve stavovém automatu

Kromě přenosové akce, akci při přechodu z jednoho stavu do druhého, lze jednotlivým stavům automatu, přiřadit tři druhy akcí: vstupní, stavovou a výstupní. Každá akce probíhá v jinou dobu, je tedy důležité jak se použijí. Příklad využití je na obr. 2.10. (stavový automat) a na obr. 2.11. (průběh časování). Je to příklad čtení dat z paměti vyňatý z CTR_RD (viz kapitola 5.4.). Ve stavu Size_addr je na adresovou sběrnici nastavena hodnota adresy dat v paměti. RD = '1' je povolení čtení z paměti. Ve stavu S_velikost jsou z datové sběrnice data uložena do proměnné Velikost. TX = '1' je povolení zápisu do UAT.

Při použití stavových akcí, viz obr. 2.10. příklad b), vypadá časování podle obr. 2.11 příklad b). Adresa v čase stavu Size_addr, je zapsána na adresovou sběrnici až ve stavu S_Velikost a platná data se objeví na datové sběrnici až v kroku následujícím. Z tohoto vyplývá, že čtená data nejsou připravena včas. Proto je nutné použít akci vstupní a výstupní, aby časování bylo správné, obr. 2.10. příklad c), a data mohla být přečtena ve stavu S_Velikost.




Obr. 2.10. Obecný popis(a)), špatné časování (b)) a správné časování (C))

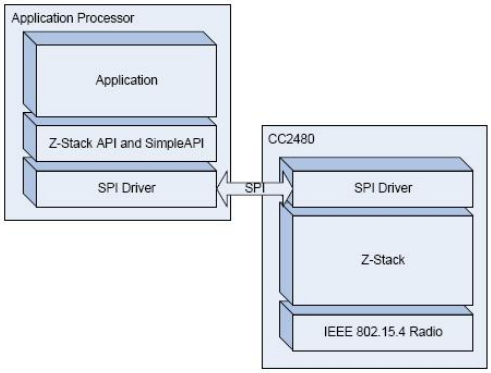



Obr. 2.11. a) špatné časování (vlevo) a b) časování správné(vpravo)

## 3.4. Transceiver CC2480

Transceiver je síťový prvek, který umožňuje překlad toku informací z jednoho typu sítě na typ jiný. Jméno tohoto prvku je odvozeno z anglických slov TRANSmitter (vysílač) a reCEIVER (přijímač). Jde o obvod, který v sobě sdružuje funkci vysílače a přijímače signálů.

Vzhledem k tomu, že I/O u FPGA je v zásadě standardní binární logika, je potřeba vyřešit nejnižší vrstvy komunikačního protokolu „off the Chip“, vrstvy MAC a PHY komunikačního protokolu ZigBee. Proto je použito transceiveru CC2480.




Obr. 2.12. Ukázka komunikace transceiveru s aplikací

### 3.4.1. Fyzické rozhraní transceiveru CC2480:

CC2480 podporuje SPI komunikační rozhraní s aplikačním procesorem (aplikaci v FPGA, dále jen hlavní procesor).

#### 3.4.1.1. SPI

SPI (Serial Peripheral Interface) je sériové synchronní periferní rozhraní. Sběrnice SPI se kvůli své implementační jednoduchosti používá v mnoha různých systémech, například pro komunikaci s některými typy pamětí EEPROM, textovými i grafickými LCD panely, A/D a D/A převodníky, hodinami reálného času (RTC) atd. Komunikace je realizována pomocí společné sběrnice. Adresace se provádí pomocí zvláštních vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení (vývody SS nebo CS).


Obr. 2.13.Sběrnice SPI: jedno řídicí (master) a jedno podřízené (slave) zařízení

**SPI konfigurace podporovaná transceiverem CC2480**

- SPI slave - vysílá podle hodinového signálu, pokud je aktivován pomocí SS/CS
- Maximální taktovací frekvence 4 MHz
- Polarita a fáze hodinového signálu :

  CPOL = 0, klidová úroveň hodinového signálu logická ‘0’.

  CPHA = 0, hodnota je čtena při vzestupné hraně
- První je nejvyšší příznakový bit MSB

**Popis signálů**

- SCK: synchronizační hodinový signál
- SS: adresace zařízení
- SI/TX: vysílání
- SO/RX: příjem

**Následující dodatečné signály**

- jsou požadované při SPI komunikaci a řízení výkonu:

- M_RDY: tento signál nastavuje FPGA při vysíláni dat do transceiveru CC2480. Je aktivní v logické ‘0’. Tento signál může být řízen nezávisle nebo je přímo zapojen na Slave Select signálu.

- S_RDY: tento signál nastavuje trasceiver CC2480, když je připraven data přijmout nebo odeslat do hlavního procesoru. Při nastavení signálu S_RDY = ‘0’ signalizuje, že je připraven na příjem dat. Při nastavení S_RDY = ‘1’ během SPI komunikace příkazu POOL nebo SREQ signalizuje, že je připraven na vyslání dat. Při SPI a AREQ příkazu signalizuje přijetí dat.

**Operace se signály**

Se signály se pracuje podle 4 následujících pravidel:

1. Hlavní procesor zahájí transakci nastavením signálu M_RDY do ‘0’ a čeká na S_RDY = ‘0’.

2. Hlavní procesor nesmí nastavit M_RDY do ‘1’ před skončením příjmu/vyslání celého rámce dat.

3. Při detekování příkazu POOL nebo SREQ, CC2480 nastaví S_RDY do ‘1’ až bude mít data

připravená pro hlavní procesor.

4. Při detekování příkazu AREQ, CC2480 nastaví S_RDY do ‘1’ až bude mít přijaty všechny bajty rámce .

**Formát vysílacího komunikačního rámce**

Formát hlavního rámce je zobrazen v následující tabulce (2.1.). Nejprve jsou přeneseny 1 a 2 bajty, tzn. levá část rámce, a poté data od nejnižšího bajtu počínaje.

| Bajty: | 1 | 2 | 0-253 |
|---|---|---|---|
| | Délka | Příkaz | Data |

Tab.2.2. Formát vysílacího komunikačního rámce

Délka: délka přenášených dat

Příkaz: nastavení přenosu

Data: přenášená data

**Nastavení přenosu :**

Pole příkaz je složeno ze dvou bajtů. Bajty jsou formátované jak je ukázáno v následující tabulce. Bajt CMD0 je přenesen jako první, poté CMD1.

| | CMD0 | | CMD1 |
|---|---|---|---|
| Bity | 7až 5 | 4 | 7 až 0 |
| | Typ | Subsystém | ID |

Tab.2.3. Soubor příkazů Command

**Type:** horní polovina bajtu může nabývat těchto hodnot:

- 0: POOL : tento příkaz je používán k získání dat z fronty. Je použitelný pouze pro přenos SPI. Tento příkaz nastavuje ID na nulu a délku dat na nulu.

- 1: SREQ : sysnchronní požadavek který vyžaduje okamžitou odezvu. např.funkce call with return (volání s návratem).
- 2: AREQ : asynchronní požadavek. Např. funkce callaback (zpětné volání) nebo funkce camm with no return (volání bez návratu).
- 3: SRSP : synchronní odezva. tento příkaz je vyslán pouze jako odezva na SREQ. Příkaz SRSP nastavuje stejné hodnoty ID jako příslušný SREQ. Délka SRSP je obecně nenulová, a tak SRSP s délkou =0 může být použit jako indikace chyby.
- 4-7 : jsou nevyužity

[7]

**Subsystém:** dolní polovina bajtu může nabývat těchto hodnot:

| hodnota | popis |
|---|---|
| 0 | nevyužit |
| 1 | SYS rozhranní |
| 2 | nevyužit |
| 3 | nevyužit |
| 4 | AF rozhranní |
| 5 | ZDO rozhranní |
| 6 | jednoduché API rozhranní |
| 7 - 32 | nevyužit |

Tab.2.4. Soubor příkazů Subsystém

**ID:**

Do proměnné Id se ukládá speciální zpráva o rozhranní.

**Reset zařízení a jeho základní nastavení - SYS_RESET_IND**

Příkaz SYS_RESER_IND je generován automaticky po resetu zařízení. K resetu zařízení a jeho nastavení je použit příkaz AREQ (viz dále) s hodnotami uvedenými v tabulce .

| 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|
| Length = 0x06 | Cmd0 = 0x41 | Cmd1 = 0x80 | Reason | TransportRev | ProductId |

| 1 | 1 | 1 |
|---|---|---|
| MajorRel | MinorRel | HwRev |

Tab.2.5. Základní nastavení transceiveru CC2480

Length : délka přenášených dat

Cmd 0,1 : nastavení přenosu

Reason : nastavení způsobu resetu (1 bajt)

- při zapnutí 0x00
- externí 0x01
- časovač 0x02

TransportRev : opakovaný přenos protokolu. Nastaven na hodnotu 2 (1 bajt)

ProdustId : ID produktu. Nastaven na hodnotu 1 (1 bajt)

MajorRel : významnější číslo verze (1 bajt)

MinorRel : menší číslo verze (1 bajt)

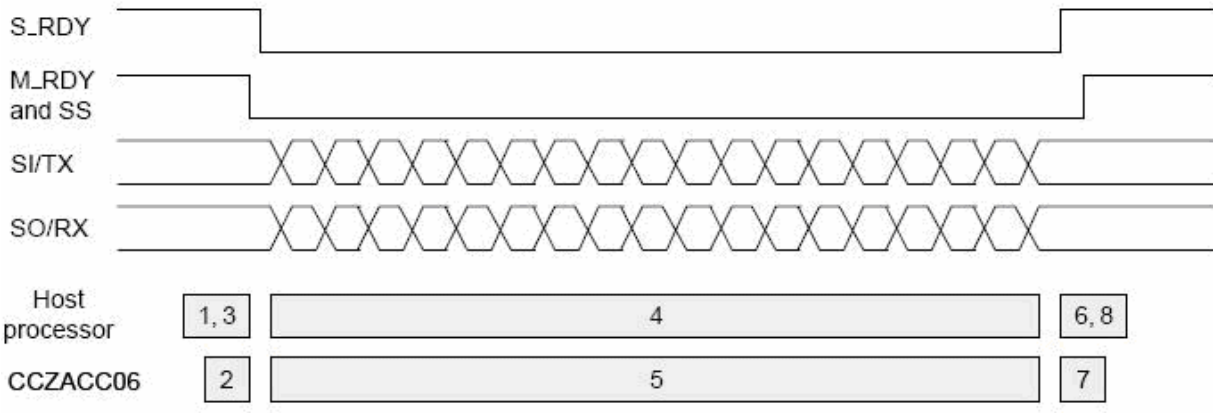
HvRev : číslo hardwaru (1 bajt)

[7]

**Popis komunikačních protokolů**

- **příkaz AREQ**

  Příkaz AREQ je použít k vysílání dat z hlavního procesoru (host procesor) do CC24800.




Obr. 2.14. Protokol AREQ

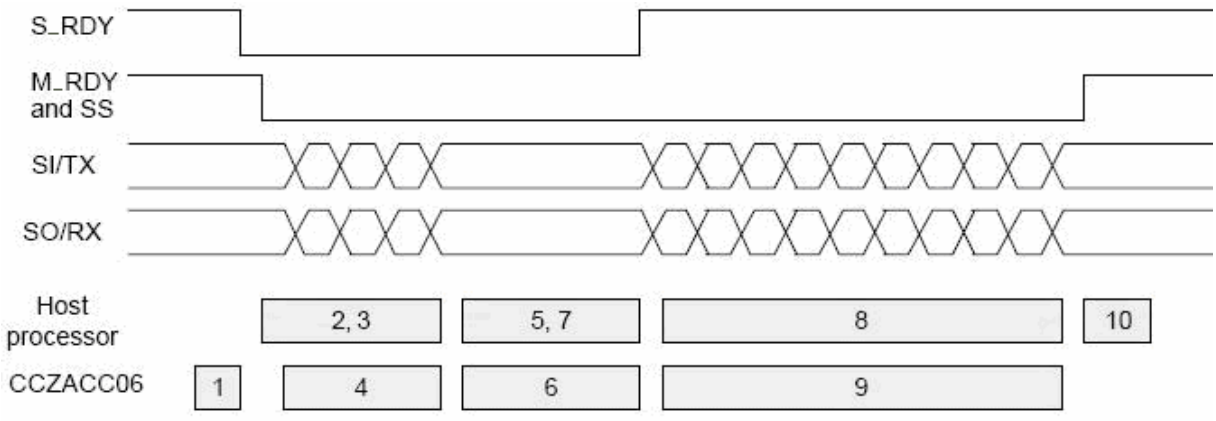
Popis sledu událostí při komunikaci, vysíláni dat a využití příkazu AREQ :

1. Vysílající hlavní procesor má data k poslání. Nastaví signál M_RDY do ‘0’ a čeká na nastavení transceiveru signálu S_RDY na ‘0’.
2. Přijímací transceiver CC2480 zjistí M_RDY = ‘0’ a nastaví S_RDY do ‘0’.
3. Hlavní procesor přečte S_RDY = ‘0’. Poté začne přenos dat.

4. Hlavní procesor vysílá, dokud není celý rámec odeslán.
5. Transceiver CC2480 přijímá, dokud není celý rámec příjmut.
6. Poté hlavní procesor čeká na S_RDY = ‘1’.
7. Transceiver. CC2480 nastaví S_RDY do ‘1’.
8. Hlavní procesor přečte signál S_RDY = ‘1’.
9. Hlavní procesor nastaví signál M_RDY do ‘1’.Konec přenosu jednoho rámce.

- **příkaz POOL**

Příkaz POOL je použit k vysílání dat směrem do hlavního procesoru (host procesor) z CC2480 transceiveru na vyžádání.




Obr. 2.15. Protokol POOL

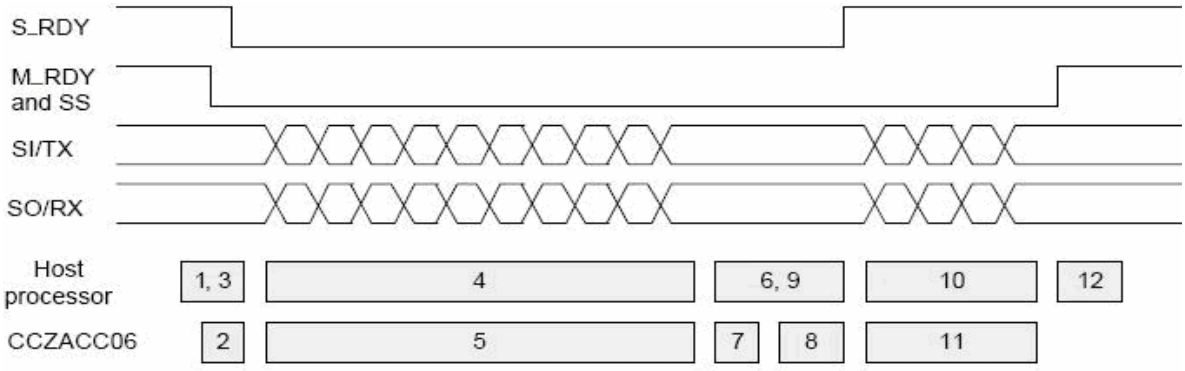
Popis sledu událostí při komunikaci, příjem dat a využití příkazu POOL

1. Transceiver CC2480 žádá o vyslání rámce pomocí příkazu AREQ. Když je připraven, nastaví signál S_RDY na ‘0’.
2. Hlavní procesor detekuje S_RDY = ‘0’ a nastaví signál M_RDY na ‘0’. Začíná vysílání dat pomocí příkazu POOL.
3. Hlavní procesor vysílá, dokud datový rámec není celý odeslán.
4. CC2480 přijímá data, dokud datový rámec není celý odeslán.
5. Hlavní procesor čeká na nastavení signálu S_RDY do ‘1’, transceiverem CC2480.
6. CC2480 připraví AREQ rámec pro vysílání. Když je připraven nastaví S_RDY = ‘1’.
7. Hlavní procesor přečte S_RDY = ‘1’ a začne data přijímat.
8. Hlavní procesor přijímá data dokud není celý rámec kompletní.

9. CC2480 vysílá data dokud není celý rámec kompletní.
10. Pokud hlavní procesor přijal kompletní rámec nastaví signál M_RDY do ‘1’. Tímto končí vysílání dat směrem do hlavního procesoru z CC2480 transceiveru na vyžádání.

- **příkaz SREQ**

Příkaz SREQ je použít ke vzdálenému volání procedur z  trasceiveru CC24800.




Obr. 2.16. Protokol SREQ

Následující sledy událostí nastanou na host procesor (FPGA) a transceiveru CC2480 a využití příkazu SREQ

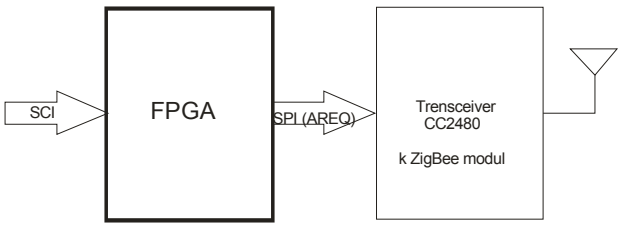
1. Vysílající hlavní procesor má data k poslání. Nastaví signál M_RDY do ‘0’ a čeká na nastavení transceiveru signálu S_RDY na ‘0’.
2. Přijímací trans. CC2480 zjistí M_RDY = ‘0’ a nastaví S_RDY do ‘0’.
3. Hlavní procesor přečte S_RDY = ‘0’. Poté začne přenos dat.
4. Hlavní procesor vysílá, dokud není celý rámec poslán.
5. Transceiver CC2480 přijímá, dokud není celý rámec poslán.
6. Poté hlavní procesor čeká na S_RDY = ‘1’.
7. Transceiver CC2480 zpracuje příkaz SREQ a spustí požadovanou funkci.
8. Transceiver CC2480 sestaví SRSP rámec. Když je připraven k vysílání nastaví signál S_RDY do ‘1’.
9. Hlavní procesor přečte S_RDY = ‘1’ a začne přijímat data.
10. Hlavní procesor přijímá data dokud není celý rámec kompletní.
11. CC2480 vysílá data dokud není celý rámec kompletní.
12. Pokud hlavní procesor přijal kompletní rámec nastaví signál M_RDY do ‘1’.

[7]

# 4. Návrh protokolu

## 4.1. Popis komunikace :

Komunikace probíhá jedním směrem a to ze sériové linky SCI přes FPGA modul do trensceiveru CC2480 ZigBee modulu. Použitý rámec přenosu je AREQ. Datové rámce procházející zařízením nejsou nijak měněna, vstupní je „ zabalen“ do výstupního pro co nejmenší poškození informace.




Obr. 3.1. Komunikace modulu FPGA se ZigBee modulem

**Parametry přenosu:**

- Rychlost vysílání 4Mb/s
- rychlost pro příjem nastavitelná dle tabulky 4.1.
- charakter přenášených dat: především data o teplotě, tlaku, váhy, EKG tedy kmitočty v rozmezí od 0,05 Hz do 100 Hz
- maximální délka přijímaného rámce 253 bajtů
- maximální délka vysílaného rámce 256 bajtů

**Přijímaný rámec:**

Formát hlavního rámce, vstupujícího do modulu FPGA ze sériové linky (SCI), je zobrazen v následující tabulce 3.1. Jako první se přijímá bajt nejnižší (tab.3.1. nejvíce vlevo), poslední nejvyšší bajt (tab.3.1. nejvíce vpravo).

| Hlavička | Identifikátor | Velikost | Data | Kontrolní součet 1.byte | Kontrolní součet 2.byte | Patička |
|---|---|---|---|---|---|---|
| 0x02 | 0xXX | 0xXX | 0xXX . . 0xXX 0xXX | CRC1 | CRC2 | 0x03 |

Tab.3.1. Formát přijímaného rámce

**Vysvětlení jednotlivých bajtů:**

- Hlavička : první vysílaný/přijímaný znak 0x02
- Identifikátor: označuje typ zprávy, která je přenášená
- Velikost rámce : velikost přenášených dat
- Data : přenášená data
- Kontrolní součty (2 bajty): celkový součet hodnot přenášených dat
- Patička : poslední vysílaný/přijímaný znak 0x03

Rezervované znaky: 0x02, 0x03, 0xF0

Náhrada: 0x02 -> 0xF0 0xFD
0x03 -> 0xF0 0xFC
0xF0 -> 0xF0 0x0F

**Vysílaný rámec:**

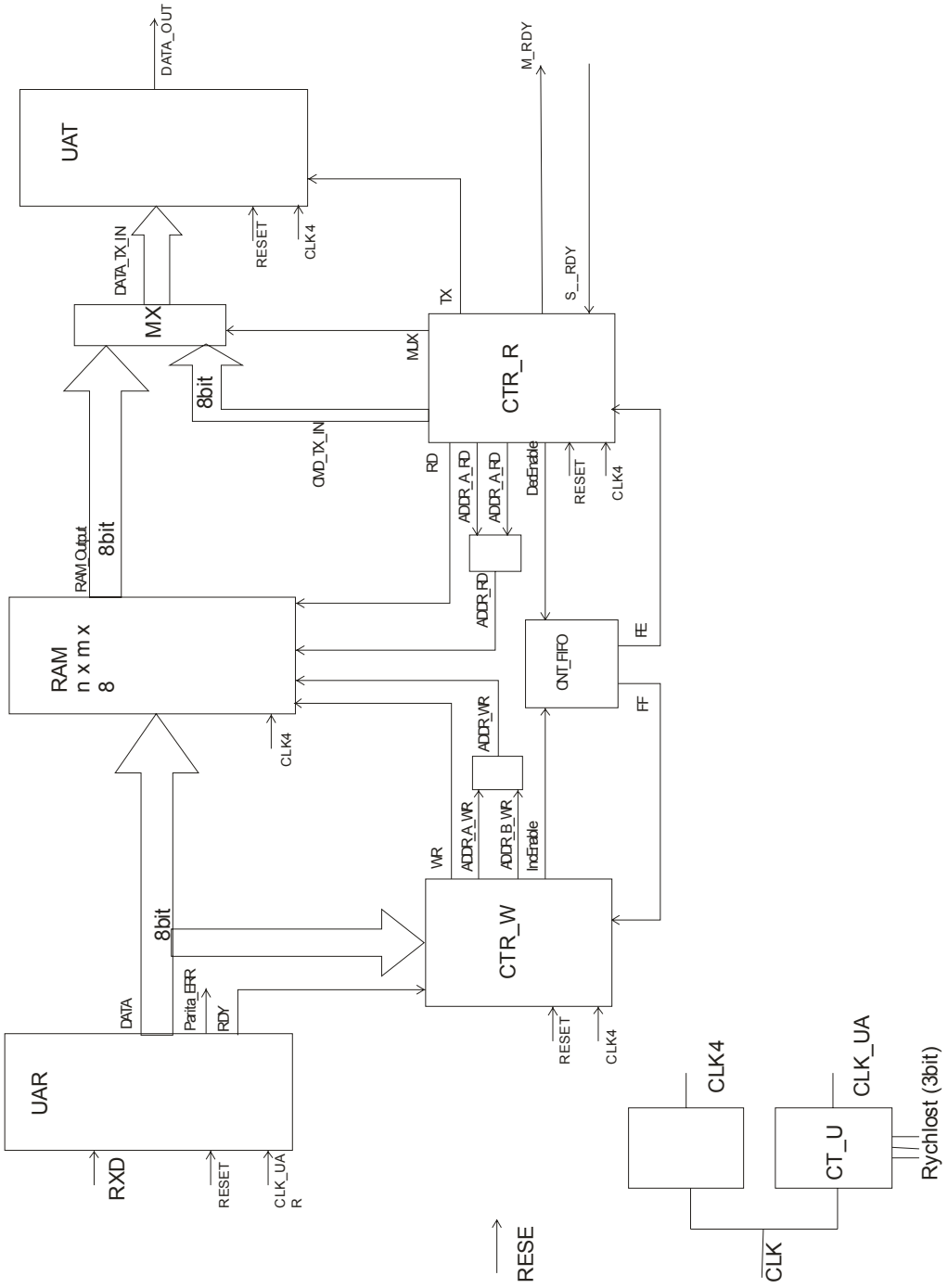
Jako formát hlavního rámce výstupu z FPGA je použit příkaz AREQ v kapitole 2.4.12 SPI.

| Bajty: | 1 | 2 | 0-253 |
|---|---|---|---|
| | Délka | Příkaz | Data |

Tab.3.2. Formát vysílaného rámce

# 5. Celkový návrh řešení

Na Obr. 4.1. je celkové schéma bloků, které jsou v této práci využity.




Obr. 4.1 Celkový návrh řešení do FPGA

**Popis jednotlivých částí :**

**UAR** – asynchronní přijímač dat. Data jsou přijímána jako sériová a rychlostí, kterou uživatel musí znát předem a nastavit jí přepínačem podle tab.4.1. Data jsou poté paralelizována a při nástupné hraně signálu RD jsou připravena na výstupu UAR.

**CTR_WR –** stavový automat příjmu dat. Data jsou při nástupné hraně signálu RDY přijímána a ukládána v blokové paměti RAM uvnitř FPGA.

**RAM –** paměť dat typu FIFO s organizací 256 x 92 x 8 bit.

**CTR_RD -** stavový automat vysílání dat a nastavování komunikace s modulem ZigBee. Při podmínce dat v paměti FIFO je zahájena komunikace s modulem ZigBee a po jeho odezvě je nastaven přenos a data připravena na adresové sběrnici Data_Tx_In z FIFO pro vysílač UAT.

**UAT –** asynchronní vysílač dat. Data jsou přijímána paralelně ze sběrnice Data_Tx_In, serializována a vysílána při hodnotě signálu TX = '1' z vysílače.

**MX –** multiplexor výběru dat z FIFO a hodnot nastavujících komunikaci s transceiverem.

**CNT_FIFO –** čítač rámců dat v paměti FIFO, nastavující signály FF a FE pro stavový automat příjmu a vysílání.

**CLK –** hlavní 50 MHz hodiny FPGA, z kterých jsou odvozeny hodiny pro UAR (CLK_UAR) a pro celou další přenosovou část (CLK4).

Dále pak jsou jednotlivé části popsány podrobněji.

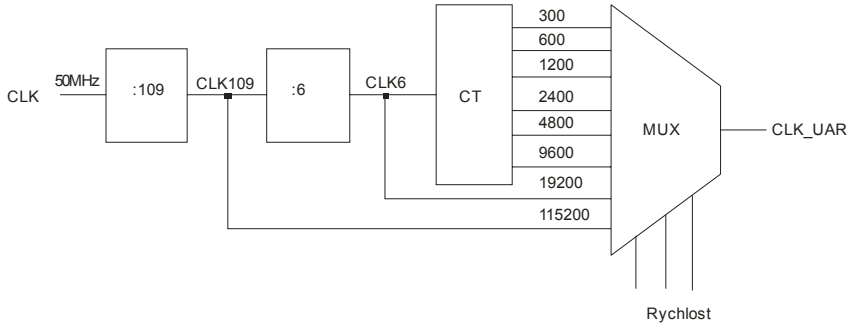
## 5.1. Časování

### 5.1.1. Časování – UAR

Pro příjem dat je nutné nastavení různých druhů rychlostí příjmu tzn., že rychlost příjmu musíme dopředu znát a musíme jej nastavit. K tomuto účelu je určen modul CT_UAR.

Signál CLK_UAR je odvozen z hlavních (CLK) 50 MHz hodin FPGA a je tvořen děličkami :109 s hodnotou výstupu 11520 Hz, :6 s hodnotou výstupu 19200 Hz a binárním čítačem CT

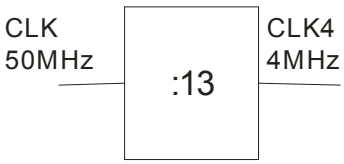
s hodnotami dvojnásobků od 300 Hz po 9600 Hz. Rychlost je vybrána multiplexorem po nastavení přepínačů rychlosti.




Obr. 4.2.Blok nastavení rychlostí příjmu

### 5.1.2. Časovaní modulu do FPGA

Celý FPGA modul (kromě UAR) je řízen max. danou rychlostí příjmu modu ZigBee – 4 MHz. Toto řeší blok CT tvořen děličkou :13, výstupním signálem CLK4.



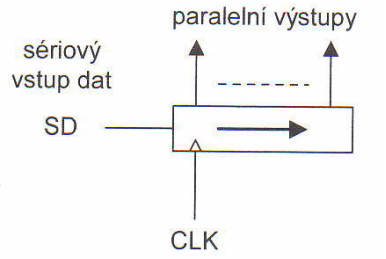

Obr. 4.3.Blok časování modulu do FPGA

Hodnoty děliček jsou celočíselné, protože v FPGA je tento blok tvořen celočíselným čítačem. Výpočet chyby přesnosti takovéto děličky je v kapitole 4.3.3. Výpočet chyby děličky.

## 5.2. Serializace a deserializace

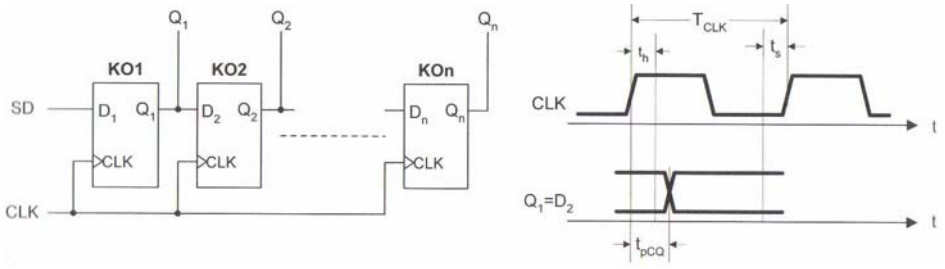
Obvod serializace je využit v části UAR a obvod deserializace v části UAT. Z těchto důvodů je datový registr popsán v této kapitole.

Spojením klopných obvodů typu D do kaskády se společným rozvodem hodinových impulzů vznikne posuvný registr. Tento blok graficky ukazuje obr. 4.4.




Obr. 4.4. Posuvný registr

Základní zapojení s časováním je zobrazeno na obr. 4.5. Hodinovým impulzem se stav na vstupu D klopného obvodu KO1 přesune na jeho výstup Q1, ale současně se původní stav na výstupu Q1 obvodu KO1 přesune na výstup Q2 obvodu KO2 atd. Výsledkem je, že celý obsah registru se posunuje doprava, zleva je doplňován vstupními daty a za posledním KO vpravo se ztrácí. Tak např. pro n = 4 ze stavu registru 0011 a vstupu 1 se přejde do stavu 1001, poslední jednička vpravo se ztrácí. Tato jednoduchá činnost však není samozřejmá a vyžaduje dodržení jistých podmínek správného časování, jak ukazuje obrázek vpravo.



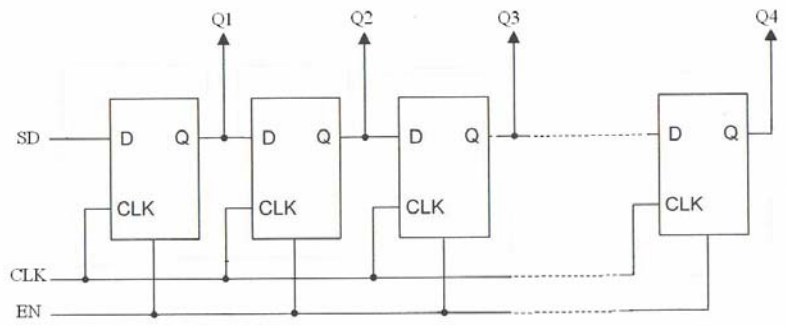

Obr. 4.5. Zapojení a časování posuvného registru

Impulz CLK je stejný pro všechny KO. Po jeho náběžné hraně se zpožděním $t_{pCQ}$ se překlopí KO1, ale pro KO2 musel být signál D2 stabilní ještě po dobu přesahu $t_h$ po hraně CLK. Dále musí být signál D2 stabilní minimálně po dobu $t_s$ před hranou dalšího CLK. Shodné je to i u dalších KO. To vede k soustavě podmínek:

$$\begin{aligned} t_{pCQ} &\geq th \\ T_{CLK} &\geq t_{pCQ} + t_s \end{aligned} \qquad (6)$$

Například pro klopný obvod 74AHCT74 s parametry $t_s > 5$ ns, $> 0$, $t_{pCQ} = 6{,}3$ ns by uvedené podmínky bylo možno splnit při $T_{CLK} \geq 11{,}3$ ns. U mnoha klopných obvodů je povoleno $t_h = 0$.

Řešením uspořádání podle předchozího obrázku je přidání povolovacího asynchronního signálu EN s trváním nejméně 1 periodou délky signálu CLK.

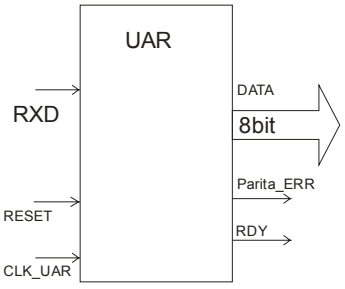



Obr. 4.6. Zapojení a časování posuvného registru s povolovacím signálem

Jedna z nejčastějších aplikací posuvného registru je převod sériového kódu na kód paralelní. Sériová data se přivádějí na vstup SD bit po bitu a vždy jsou doprovázena signálem CLK. Po patřičném počtu impulzů jsou paralelní data k dispozici na výstupech Q. Opačná aplikace, tj. převod paralelních dat na data sériová, vyžaduje zaplnění posuvného registru daty a následné přivedení patřičného počtu impulzů CLK. Sériová data jsou k dispozici na posledním KO vpravo.[1]

## 5.3. UAR:

- asynchronní přijímač dat. Tento blok zobrazený na obr. 4.7. přijímá sériová data ze sériové linky RXD rychlostí, kterou uživatel musí znát předem a nastavit jí přepínačem podle tab.4.1. Data jsou poté paralelizována a při nástupné hraně signálu RD = ‘1’ jsou připravena na výstupu UAR. Dalším z výstupních signálů je Paritta_Err, který signalizuje správnost přijetí dat kontrolním součtem celého bajtu. Parita je nastavitelná na sudou nebo lichou.




Obr. 4.7. Asynchronní přijímač dat

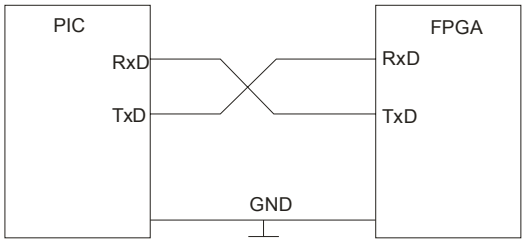
### 5.3.1. SCI komunikace

SCI je univerzální asynchronní sériové komunikační rozhraní. SCI nepoužívá ke komunikaci hodinovou linku, má jen linku datovou, kde jeden vývod je využit pro vysílání a druhý pro příjem. Obě operace mohou probíhat nezávisle na sobě. V plně duplexním režimu probíhají zároveň. Nejčastější použití SCI je pro komunikaci s PC prostřednictvím sériového portu a protokolu RS-232.

V této diplomové práci se řeší část příjmu z linky SCI, právě z RS-232.

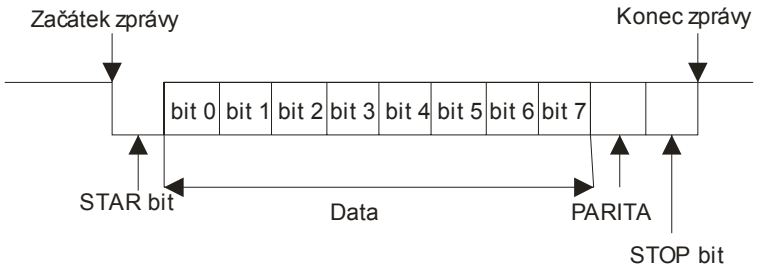
#### 5.3.1.1. Popis komunikace:

Při „ klidu“, tj. v době kdy se nic nevysílá, nepřijímá, je na lince logická ‘1’. Při vysílání se na datové lince objeví logická ‘0’ po určitou dobu představující začátek vysílání. Tento pokles (z logické ‘1’ do logickou ‘0’) při vysílání je synchronizací s příjemcem pro příjem dat. Tento signál zůstane na nízké úrovni během celého start bitu a dále je jeho úroveň podle posílaných dat. První se posílá bit nejnižší LSB, poslední nejvyšší bit MSB je následován stop bitem, který je opět v logické ‘1’.




Obr. 4.8. Příklad komunikace SCI mezi procesorem a FPGA

**Formát SCI zprávy vypadá takto:**

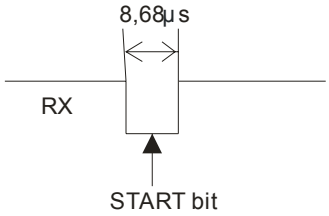



Obr. 4.9. Formát obecná SCI zprávy

Celkem je vysláno/přijato 11 bitů během přenosu. Jako první bit je vyslán/přijat START bit, logická ‘0’ na lince. Dále pak 8 bitů dat, paritní bit ( PARITA), a jako poslední je vyslán/příjmut STOP bit, logická ‘1’ na lince. Ve stavu klidu (nic se nevysílá/přijímá) je linka ve stavu logické ‘1’.

### 5.3.2. Nastavení přenosové rychlosti příjmu z SCI

Jelikož u SCI přenosu nejsou využívány žádné synchronizační hodiny, musí obě zařízení vědět o přenosové rychlosti dopředu. Vysílací zařízení udává tuto rychlost a u FPGA se nastavuje fyzicky pomocí přepínače, viz tab. 4.1. Zde je využita rychlost maximální 115200 bd. (vztahuje se i k výpočtům – obr. 4.10.).



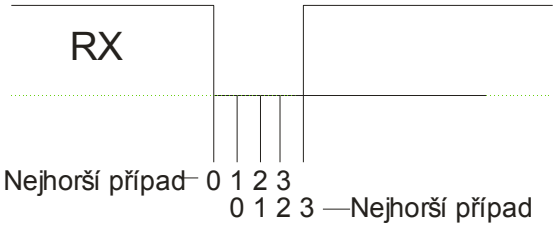

Obr. 4.10. Délka trvání ($t_{vzork}$) 1 bit (např: START bit)

### 5.3.3. UAR - Výpočet hodnot vzorkování, děličky a chyby dělení

Základními parametry pro výpočty je frekvence krystalu (50 MHz) použitého u FPGA viz kapitola FPGA a rychlost přenosu u SCI komunikace. U vzorkovací frekvence je použita mocnina dvou $2^n$, kde n je celé kladné číslo. Toto číslo je použito z důvodu použití binární logiky čítačů.

**Vzorkování přijímaného bitu:**

Na obr. 4.11. je zobrazen nejhorší možný případ zjištěných vzorků. Jedná se o nultý a třetí vzorek. Ty, totiž mohou být odečítány v době, kdy se vzorkovaný bit mění z logické ‘0’ do ‘1’ a naopak. Není tak zaručený správný vzorek přijímaného bitu . S těmito vzorky se proto vůbec nepočítá.




Obr. 4.11. Nejhorší případ vzorkování 1 bitu (zde logická ‘0’) linky příjmu RX

$v_{př}$ – rychlost přenosu
$n_{vzork}$ – počet vzorků
$n_{bit}$ – počet bitů

**Výpočet frekvence vzorkování:**

$$f_{vzork} = v_{př} \cdot n_{vzork} \tag{7}$$

**Výpočet doby jednoho vzorku:**

$$t_{vzork} = \frac{1}{n_{vzork} \cdot v_{př}} [\mu s] \tag{8}$$

**Výpočet hodnoty děličky:**

$$Hodnota = \frac{50000000}{f_{vzork}} \tag{9}$$

**Výpočet chyby děličky:**

Jelikož při použití vnitřního krystalu Spartanu-3 o frekvenci 50 MHz nelze dělením dosáhnout přesné vzorkovací frekvence, musíme se spokojit z určitou chybou $\Delta f_{vzork}$. Tato chyba vzniká zaokrouhlením hodnoty děličky na celé číslo a nesmí být tak velká, aby ovlivnila správnost přijatých dat. Je vztažena k přenosu přijatých 11 bitů a rychlosti přenosu, poté se příjem opět synchronizuje.

**Čas příjmu :**

$$t_{rx} = \frac{1}{v_{př}} \cdot n_{bit} [\mu s] \tag{10}$$

**Čas příjmu po použití děličky a vzorkování:**

$$t_{rxD} = \frac{n_{vzork} \cdot Hodnota}{50000000} \cdot n_{bit} [\mu s] \tag{11}$$

**Výpočet chyby Δ :**

$$\Delta = t_{rx} - t_{rxD} \ [\mu s] \tag{12}$$

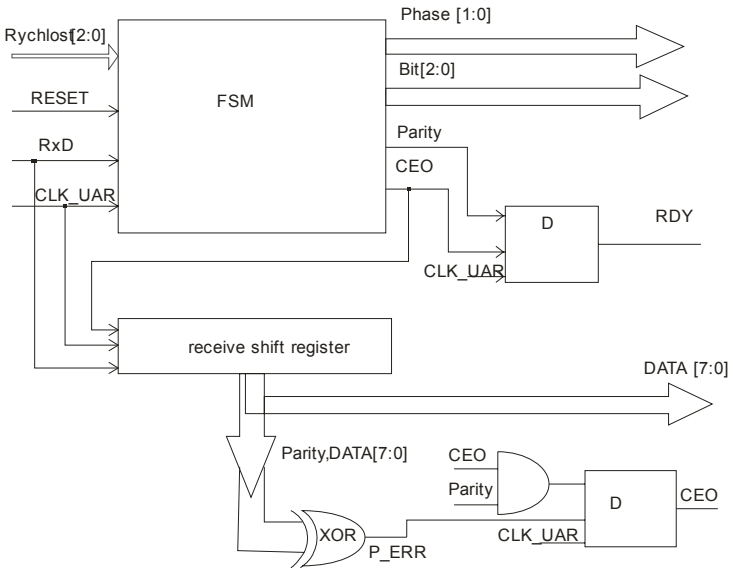
- Frekvence krystalu: f = 50 MHz
- Mocnina: n = 2 => počet vzorků = 4
- Přijímáno 11 bitů.

| pozice přepínače | rychlost přenosu [bit/s] | vzorkování f[Hz] | Čas příjmu 11 bitů [μs] | hodnota děličky (zaokrouhleno nahoru) | Čas příjmu 11 bitů [μs] /dělička/ | Δ [μs] | Absolutní chyba [%] |
|---|---|---|---|---|---|---|---|
| 000 | 115200 | 460800 | 95,486 | 109 | 95,9200 | 0,4339 | 0,4523 |
| 001 | 19200 | 76800 | 572,917 | 652 | 573,7600 | 0,8433 | 0,1470 |
| 010 | 9600 | 38400 | 1145,833 | 1303 | 1146,6400 | 0,8067 | 0,0704 |
| 011 | 4800 | 19200 | 2291,667 | 2605 | 2292,4000 | 0,7333 | 0,0320 |
| 100 | 2400 | 9600 | 4583,333 | 5209 | 4583,9200 | 0,5867 | 0,0128 |
| 101 | 1200 | 4800 | 9166,667 | 10417 | 9166,9600 | 0,2933 | 0,0032 |
| 110 | 600 | 2400 | 18333,333 | 20834 | 18333,9200 | 0,5867 | 0,0032 |
| 111 | 300 | 1200 | 36666,667 | 41667 | 36666,9600 | 0,2933 | 0,0008 |

Tab.4.1. Hodnoty rychlostí, vzorkování, děliček, času příjmů a chyby

### 5.3.4. Architektura UAR :

Architektura přijímače dat (UAR) je na obr. 4.11. Celý příjem řídí modul FSM na základě vstupních hodin CLK_UAR. Vstupní sériová data jsou paralelizována (Shift registr) a připravena při signálu RDY na výstupní sběrnici DATA [7:0].



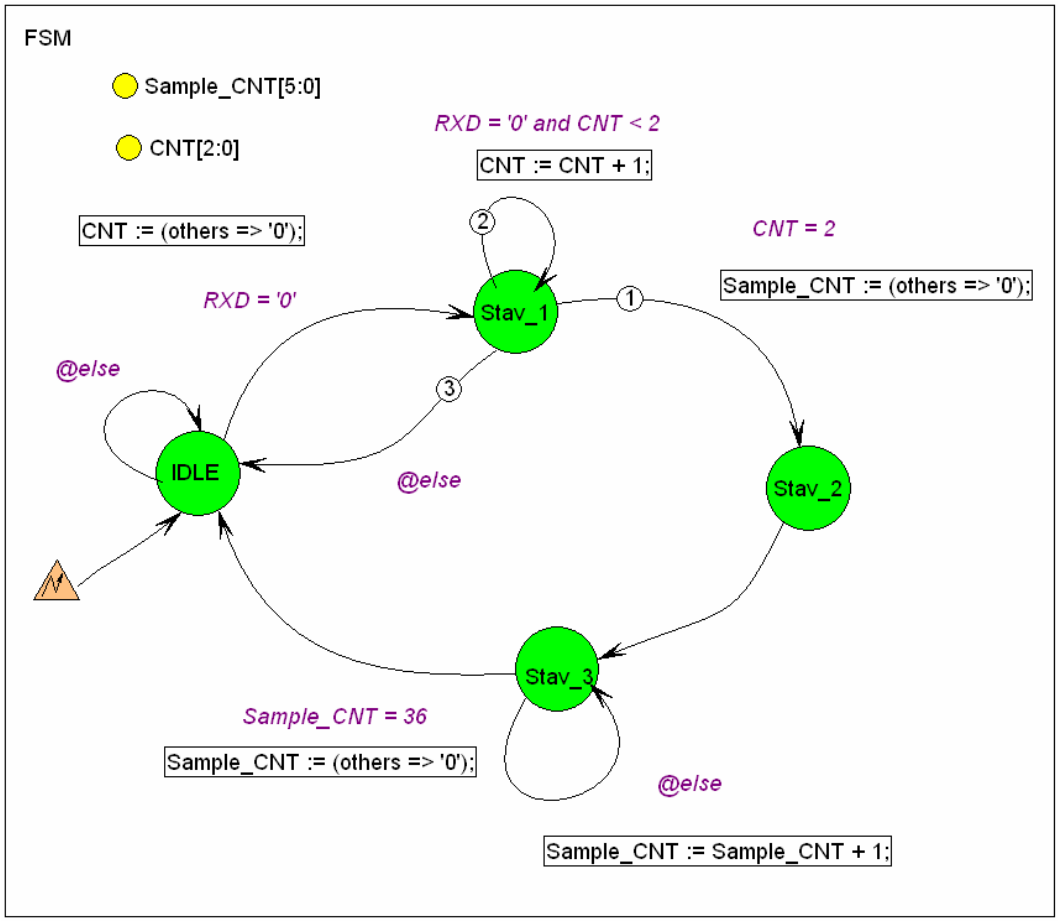

Obr. 4.11. Architektura UAR

**Vstupy, výstupy:**

```
port (
    -- vstupní signály
       CLK          : in  STD_LOGIC;
       RESET        : in  STD_LOGIC;
    -- vstupní data
       RXD          : in STD_LOGIC;
    -- výstupní signály
       RDY          : out std_logic;
       PARITA_ERR: out std_logic
    -- výstupní data
       DATA         : out std_logic_vector (7 downto 0);
       S_BIT        : out STD_LOGIC_VECTOR (2 downto 0);
       PHASE        : out STD_LOGIC_VECTOR (1 downto 0);
       );
```

### 5.3.5. UAR - Stavový automat FSM

Řeší část příjmu asynchronních sériových dat z linky RX portu RS232.




Obr. 4.12. Stavový automat FSM pro UAR

#### 5.3.5.1. Popis automatu FSM:

Po detekci prvního vzorku rovném logické ‘0’, stav IDLE možného START bitu na vstupu RxD, se automat překlopí do dalšího stavu Stav_1, kde je testován tento bit ještě 2x. Pokud je druhý vzorek také roven nule, je tento bit považován za START bit. Při druhém vzorku RxD = ‘1’ se vracíme na začátek testování. Stav_2 automat prochází, protože nás 4. vzorek nezajímá (označen x v tab.4.2.) . Ve Stav_3 se dopočítává zbylých 36 vzorků, včetně parity. STOP bit nás nezajímá. Poté (po příjmu bajtu dat) se vracíme k opětovnému testování linky RxD. Rozpis vzorků, které nás zajímají (vyznačených tučně) a vzorků, které nás nezajímají (vyznačených písmenem x ) je v tabulce 4.2.

| klid na lince | | | | | | START bit | | | | DATA | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | x | 0 | **0** | 0 | x | x | **I** | I | x | x | **0** | 0 | x | x | … |
| | | | | | | 0 | | | | 1 | | | | 0 | | | | | |

Tab.4.2. Popis přijímaných vzorků

### 5.3.6. Popis dalších částí UAR:

Po detekci START bitu se další data bit po bitu přesunou do registru pro příjem (receive shift register). Zápis do registru je povolen při hodnotě výstupního signálu PHASE čase Sample_CNT(1 downto 0) = ‘01‘, což znamená zápis při každém 4. vzorku. Po naplnění celého registru je vypočtena parita (P_ERR <= Data_reg (8) xor Data_reg (8) xor … xor Data_reg (0);) signalizující správné přijetí dat. Výstup z hradla typu D (RDY) signalizuje přijetí všech 8 bitů dat.

Všechny obvody jsou synchronizovány s hodinami CLK_UAR.

### 5.3.7. Vzorkování s větší periodou a odečet tří vzorků

Pro lepší a hlavně přesnější zjištění vzorku, je vstupní signál navzorkován s větší vzorkovací frekvencí.

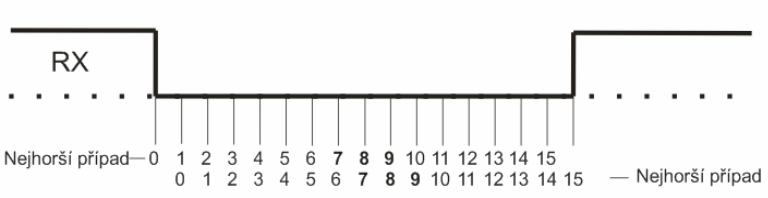
- Frekvence krystalu: f = 50 MHz
- Mocnina: n = 4 => počet vzorků = 16
- Přijímáno 11 bitů.

| pozice přepínače | rychlost přenosu [bit/s] | vzorkování f[Hz] | Čas příjmu 11 bitů [μs] | hodnota děličky (zaokrouhleno nahoru) | Čas příjmu 11 bitů [μs] /dělička/ | Δ [μs] | Absolutní chyba [%] |
|---|---|---|---|---|---|---|---|
| 000 | 115200 | 1843200 | 95,486 | 28 | 98,5600 | 3,0739 | 3,1188 |
| 001 | 19200 | 307200 | 572,917 | 163 | 573,7600 | 0,8433 | 0,1470 |
| 010 | 9600 | 153600 | 1145,833 | 326 | 1147,5200 | 1,6867 | 0,1470 |
| 011 | 4800 | 76800 | 2291,667 | 652 | 2295,0400 | 3,3733 | 0,1470 |
| 100 | 2400 | 38400 | 4583,333 | 1303 | 4586,5600 | 3,2267 | 0,0704 |
| 101 | 1200 | 19200 | 9166,667 | 2605 | 9169,6000 | 2,9333 | 0,0320 |
| 110 | 600 | 9600 | 18333,333 | 5209 | 18335,6800 | 2,3467 | 0,0128 |
| 111 | 300 | 4800 | 36666,667 | 10417 | 36667,8400 | 1,1733 | 0,0032 |

Tab.4.3. Hodnoty rychlostí, vzorkování, děliček, času příjmů a chyby

Při větší vzorkovací frekvenci může dojít ke zjištění nesprávné hodnoty (z důvodu rušení na lince RxD), proto jsou vyčteny tři vzorky a z nich jsou vybrány 2 převažující. V tomto případě jsou vybrány 7., 8. a 9. vzorek, ze kterých se vybírají ty převažující.

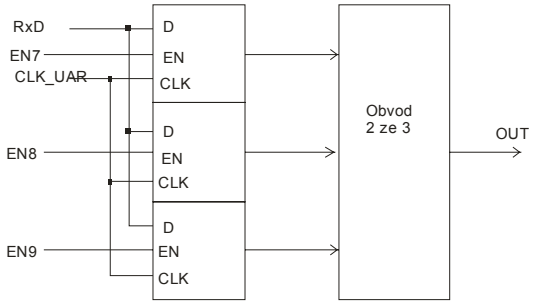
**Vzorkování přijímaného bitu:**




Obr. 4.13. Nejhorší případ vzorkování 1bitu ( zde logická ‘0’ ) linky příjmu RX

**Výběr vzorku:**

Obvod pro výběr vzorku je na obr. 4.15. Vzorek 7., 8. a 9. jsou postupně ukládány do klopných obvodů typu D. Poté jsou pomocí obvodu „ 2 ze 3“ vybrány minimálně dva stejné vzorky, viz tabulka 4.4., a tato převažující hodnota je vyvedena na výstup.




Obr. 4.15. Obvod pro výběr vzorků

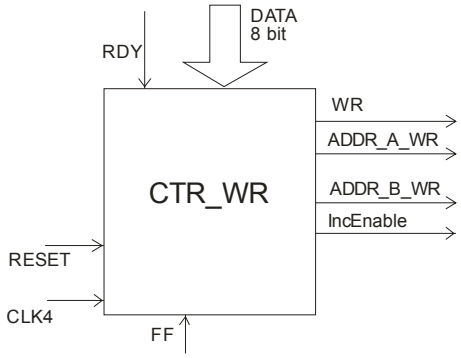
| 7 | 8 | 9 | OUT |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Tab.4.4. Tabulka výběru vzorků

## 5.4. CTR_WR - Controller Write

Controller Write je stavový automat kombinací Moorova a autonomního automatu. Řídí příjem paralelních dat z UAR při náběžné hraně signálu RDY. Data jsou 8-mi bitová, oddělena od Start bitu, paritního bitu a Stop bitu. Tato data nám udávají hodnotu bajtů, které zastupují: Hlavičku, Identifikátor, Velikost, Data, dva Kontrolní součty a Patičku (viz Tabulka 3.1.)

Tento stavový automat je podobný rouře, která postupně přijímá bajt po bajtu a zapisuje je do paměti FIFO, než příjme celý rámec, což je odvozeno od bajtu Velikosti dat. Poté se přijímá další rámec, opět po jednotlivých bajtech, než je paměť plná. Tento stav indikuje signál FF. Pokud je aktivní nepřijímají se žádná data.




Obr. 4.16. Stavový automat příjmu dat – CTR_WR

**Vstupy, výstupy**

```
port(
   -- vstupní data
        DATA_IN     : in std_logic_vector (7 downto 0);
   -- vstupní signály
        RDY         : in std_logic;
        CLK4        : in std_logic;
        RESET       : in std_logic;
```

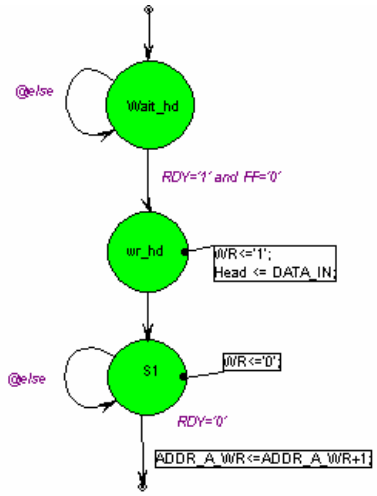
```
        FF              : in std_logic;
-- výstupní adresy (horní a dolní část)
        Addr_A_Wr       : out STD_LOGIC_VECTOR (7 downto 0);
        Addr_B_Wr       : out STD_LOGIC_VECTOR (7 downto 0);
-- výstupní signály
        WR              : out std_logic;
        IncEnable       : out std_logic;
         );
```

### 5.4.1. CTR_WR - Stavový automat FSM

Stavový automat CTR_WR je rozdělen do dvou částí:

**Část příjmu informačních bajtů**

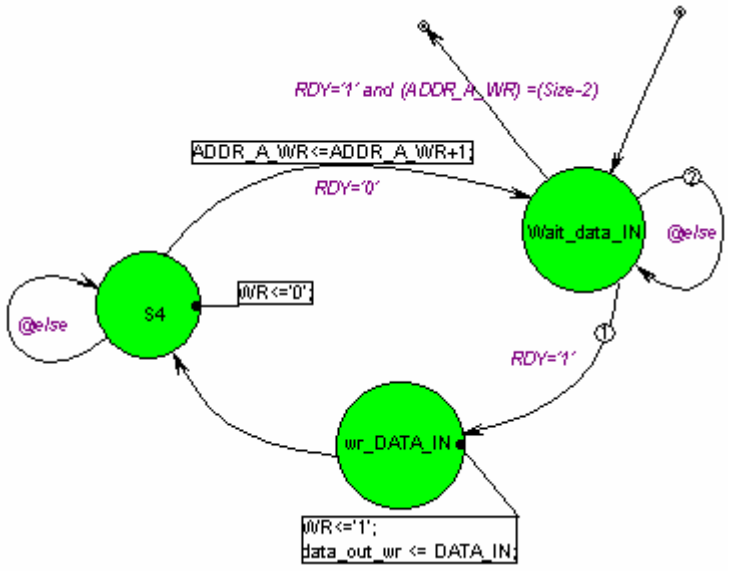
Informačními bajty jsou Hlavička, Identifikátor, Velikost, Data, dva Kontrolní součty a Patička. Při příjmu prvního bajtu automat zjišťuje ve stavu Weit_, zda není paměť FIFO plná (FF = logická ‘1’) a jestli jsou data připravena ke čtení na výstupní datové sběrnici bloku UAR (při RDY = logická ‘1’). Pokud ano, tak ve stavu WR_ povolí zápis dat do paměti (signál WR <=‘1’;) po dobu jedné periody hodin. V následujícím stavu S nastaví všechny tyto signály zpět na původní hodnoty (logickou ‘0’) a přičte spodní část adresy Addr_A_Wr o jedničku. Při příjmu posledního informačního bajtu (Patička) je přičtena horní část adresy Addr_B_Wr a nastaven signál IncEenable na jednu periodu CLK4 do jedničky. V tuto chvíli se přičítá hodnota čítače paměti CNT_FIFO. Na obrázku 4.17. je zobrazena část příjmu bajtu Hlavičky.




Obr. 4.17. Příjem bajtu Hlavička

**Část příjmu datových bajtů**

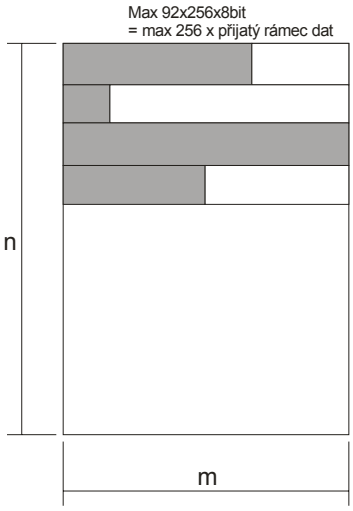
Tato část bajtů, maximálně 250 do maximální velikosti rámce, je podmíněna hodnotou Size a uložena v části příjmu informačních bajtů - Velikost. Po tuto hodnotu bude stavový automat Ctr_Wr přijímat v cyklu data, dokud se nebude adresa Addr_A_Wr posledního bajtu rovnat hodnotě (Velikost – 2). Systém načtení dat z UAR a zápisu do FIFO je stejný jako u části příjmu informačních bajtů.




Obr. 4.18.Příjem bajtů dat

## 5.5. RAM:

Jako mezičlánek mezi čtením z SCI a vysíláním přes SPI je využita jako vyrovnávací, synchronní paměť RAM s konstrukcí paměti FIFO. Velikost paměti je určena počtem a maximální velikostí přijímaného rámce, je tedy s organizací 92 x 256 x 8 bit ( využití 188416 bitů blokové paměti Spartanu 3)



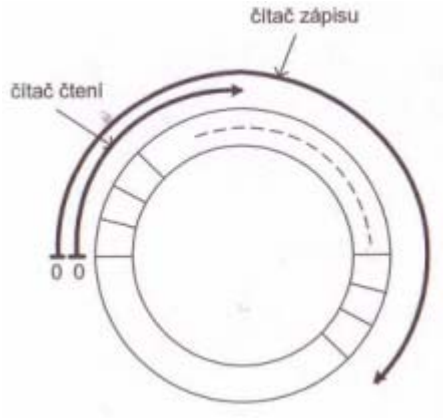

Obr. 4.18. Organizace paměti RAM

Je založena na zjednodušené dvojbránové paměti a dvou ukazatelích na adresy, realizovaných čítači. Ukazatelé se skládají ze dvou adres: horní (počet rámců) a dolní části (velikost rámců).

Addr_WR <= Addr_A_Wr & Addr_B_Wr;

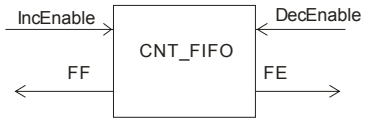
Addr_RD <= Addr_A_Rd & Addr_B_Rd;

Zjednodušení DPM spočívá v omezení funkce u obou bran – jedna dovoluje jen zápis a druhá jen čtení. Po každém zápisu dat se posune čítač zápisu a po každém čtení se posune čítač čtení. Vzhledem k funkci čítačů, kdy po nejvyšší adrese následuje nula, lze adresy v paměti znázornit jakoby navinuté na kružnici.




Obr. 4.19. Kružnice čtení a zápisu

Čítač zápisu obsahuje většinou vyšší číslo než čítač čtení. To odpovídá částečnému zaplnění FIFO. Jakmile by čítač čtení „doběhl“ čítač zápisu, odpovídalo by to úplnému vyprázdnění FIFO. Naopak čítač zápisu nesmí „předběhnout“ čítač čtení, což by odpovídalo úplnému zaplnění FIFO. Obě adresy jsou proto porovnávány v komparátoru CNT_FIFO, který v případě zaplnění nebo vyprázdnění FIFO zablokuje patřičný čítač a tím i příjem nebo vysílání dat signály FF nebo FE.




Obr. 4.19. Čítač dat v paměti FIFO

**Vstupy, výstupy FIFO**

```
port(
   -- vstupní data
        DATA_IN     : in std_logic_vector (7 downto 0);
   -- vstupní signály
        CLK4        : in std_logic;
        WR          : in std_logic;
        RD          : in std_logic;
  -- vstupní adresy
        Addr_Wr     : in STD_LOGIC_VECTOR (7 downto 0);
        Addr_Rd     : in STD_LOGIC_VECTOR (7 downto 0);
   -- výstupní data
        Ram_output  : out std_logic_vector (7 downto 0);
        );
```

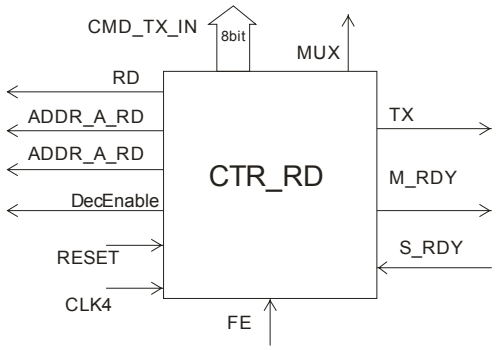
**Vstupy, výstupy CNT_FIFO**

```
port(
     -- vstupní signály
        IncEnable           : in std_logic;
        DecEnable           : in std_logic;
     -- výstupní signály
        FF                  : out std_logic;
        FE                  : out std_logic;

        );
```

## 5.6. CTR_RD – Controller Read:

Controller Read je stavový automat kombinací Moorova a autonomního automatu. Řídí nastavování modulu ZigBee a vysílání sériových dat z UAT. Nastavení tohoto modulu je provedeno po resetu, tzn. ihned po zapnutí zařízení. Vysílání se provádí podle hlavního vysílacího rámce a příkazu AREQ. Vysílání je podmíněno alespoň jedním rámcem přijatých dat v paměti, tedy CNT_FIFO > 0. Celkový popis je zobrazen ve FSM CTR_RD, rozdělen do částí tak, jak jdou za sebou.




Obr. 4.20.Controller Read

**Vstupy, výstupy**

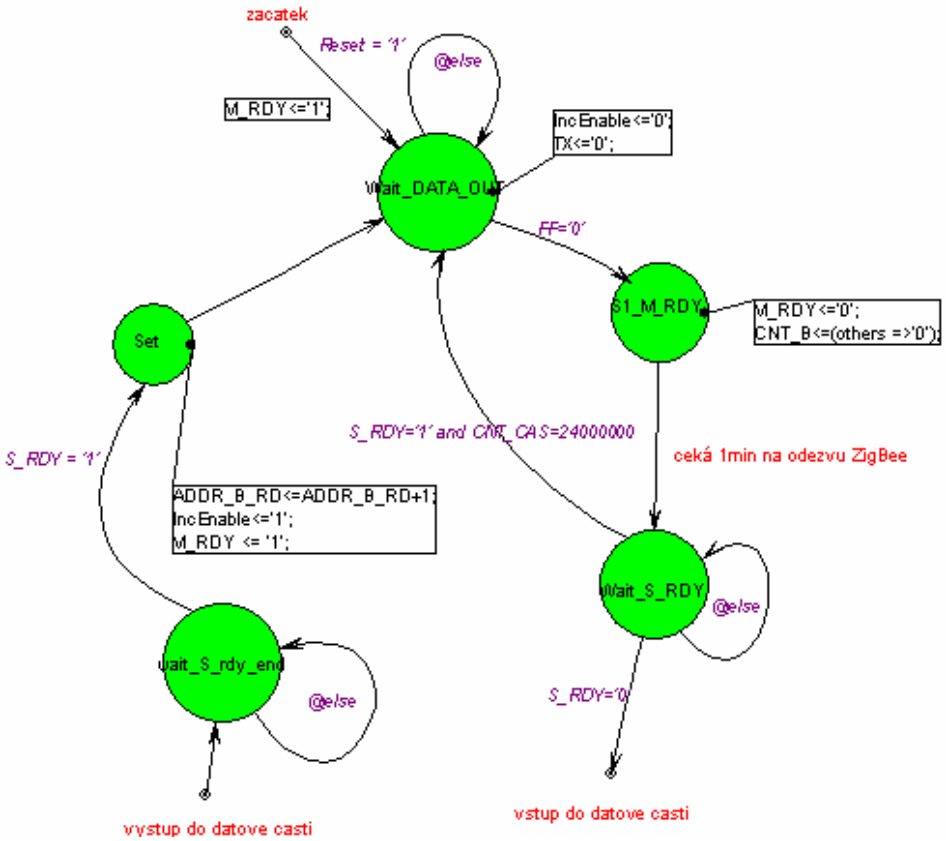
```
port(
   -- vstupní signály
        S_RDY        : in std_logic;
        CLK4         : in std_logic;
        RESET        : in std_logic;
        FE           : in std_logic;
   -- výstupní data
        CMD_TX_IN :out STD_LOGIC_VECTOR (7 downto 0);
   -- výstupní adresy
        Addr_A_RD   : out STD_LOGIC_VECTOR (7 downto 0);
        Addr_B_RD   : out STD_LOGIC_VECTOR (7 downto 0);
   -- výstupní signály
        M_RDY        : out std_logic;
        TX           : out std_logic;
        RD           : out std_logic;
        DecEnable    : out std_logic;
        Mux          : out std_logic;
          );
```

### 5.6.1. CTR_RD - Stavový automat FSM

**Část nastavení komunikace s transceiverem CC2480**

První spuštění tohoto automatu je po resetu celého FPGA modulu, kdy se zahájí komunikace s transceiverem CC2480, nastavením signálu M_RDY do logické ‘0‘. Pokud ovšem tato část už proběhla, předchází nastavení signálu M_DRY kontrola dat v paměti, zda v ní jsou (signál FE). Po „ shození“ signálu M_RDY, čekáme na odezvu transceiveru signálem S_RDY nastaveným do logické ’0‘, tím zajistíme propojení mezi FPGA a transceiverem. Pokud se tak nestane do jedné minuty, cyklus se opakuje. Následuje datová část vysílání dat do UAT.

Po vyslání všech dat, buď z paměti FIFO nebo konstant k nastavení přenosu, nastaví transceiver signál S_RDY do logické ‘1’ a FPGA na toto reaguje nastavením signálu M_RDY také do logické ‘1’. Pokud proběhla datová část, z FIFO je přičtena adresa vyslaného rámce ADDR_B_RD s kontrolou počtu rámců v paměti. Pokud proběhla část konstant, k nastavení přenosu následuje zopakování zahájení komunikace a část datová.

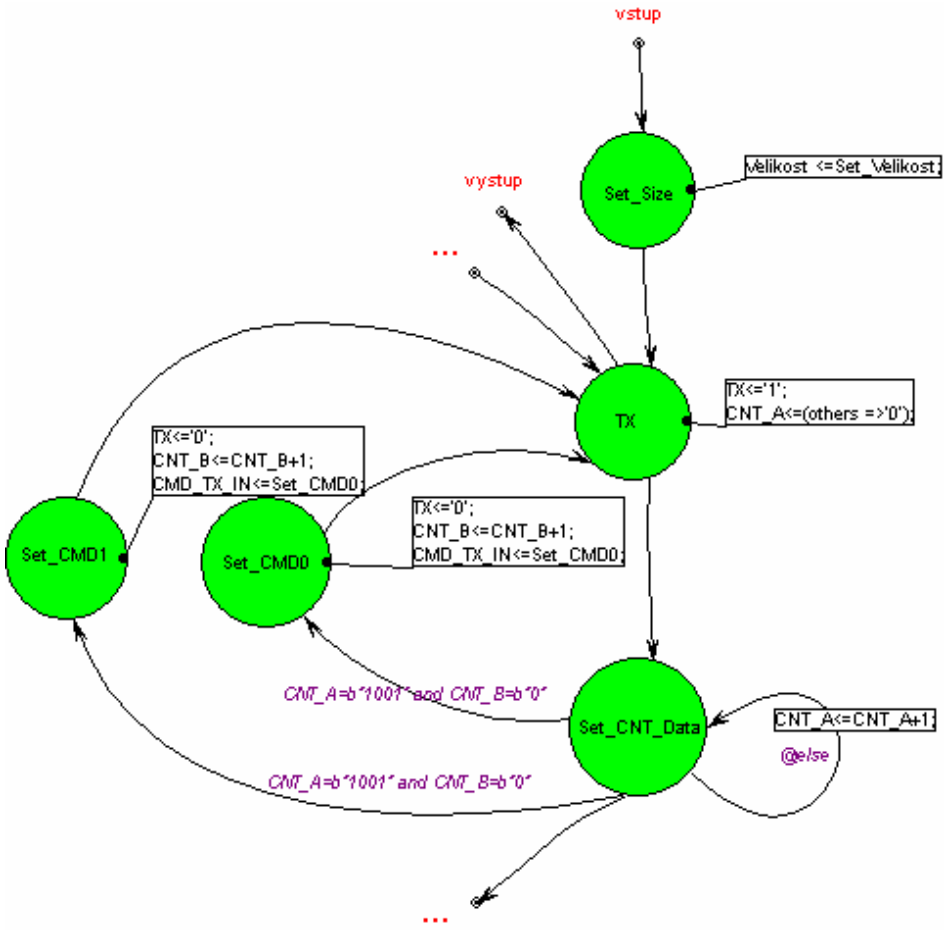



Obr. 4.21. Nastavení komunikace s transceiverem CC2480

**Část nastavení transceiveru CC2480**

Pro nastavení transceiveru je nutné znát dopředu celkovou velikost vysílaných dat. Jednotlivé nastavující parametry a tím i jejich velikost je v příloze Datasheetu CC2480. Po navázání komunikace jsou postupně vysílány 3 hlavní bajty (Velikost, CMD0, CMD1) a poté další, které doplňují nastavovací parametry (viz tabulka 2.5.). Dle konstanty Velikost se provede vyslání jednotlivých bajtů, než je celý rámec odeslán.

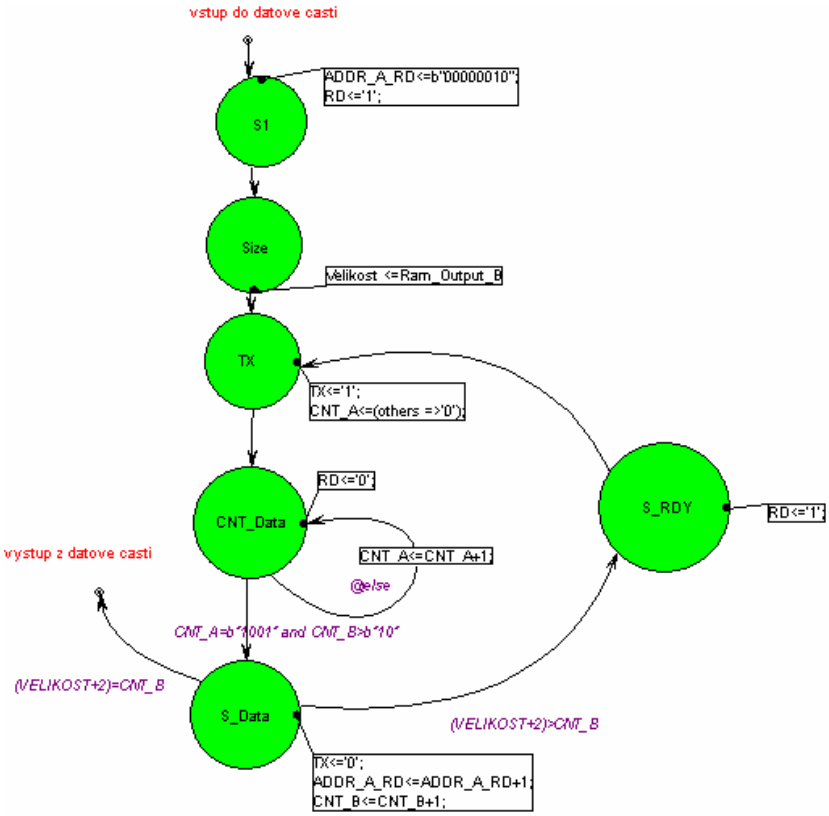
Důležitou částí tohoto FSM je čítač, který udává délku času 11 CLK4 impulsů, tj. dobu signálu TX v logické ‘1’ pro UAT, který serializuje vysílaný bajt dat.




Obr. 4.22. Nastavení transceiveru CC2480

**Část vysílání dat**

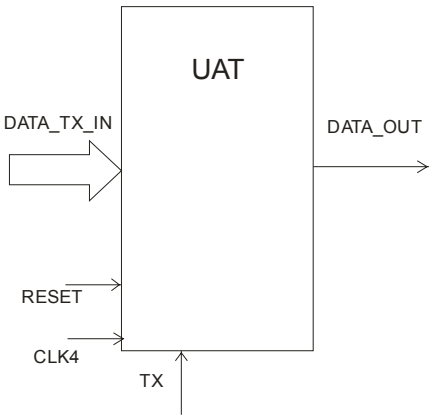
Po nastavení přenosu, jak z hlediska komunikace, tak z nastavení samotného transceiveru, je nutné zjistit dopředu celkovou velikost vysílaných dat. Ta, je uložena na statické adrese ve FIFO. Lze ji načíst jen při správném časování stavového automatu (viz obrázek 2.10. a 2.11.) tak, aby byla platná data na datové sběrnici včas. Dále se střídají procesy nastavování adresové sběrnice FIFO pro čtení dat s čítačem vyslaných bajtů a čítačem, který udává opět délku času 11 CLK4 impulsů ( TX v logické '1') pro UAT, který serializuje vysílaný bajt dat, dokud není celý rámec odeslán.




Obr. 4.23 Vysílání dat – CTR_RD

## 5.7. UAT:

Univerzální asynchronní vysílač dat umožňuje vysílání dat, pokud jsou v paměti a jsou povolena automatem CTR_RD a signálem TX. Data jsou serializována (viz kapitola 4.2). Vysílá se 11 bitů (viz komunikace 4.3.1.1), tzn. po dobu 11 CLK4. Synchronní je s hlavními hodinami CLK.




Obr. 4.24. Vysílač dat UAT

```
port(
   -- vstupní data
        Data_Tx_In          : in std_logic_vector;
   -- vstupní signály
        CLK4                : in std_logic;
        RESET               : in std_logic;
        TX                  : in std_logic;
   -- výstupní data
        Data_Out            : out std_logic;
```

# 6. Experimentální testování

Testování je provedeno v simulátoru Test bench waveform, komponenty programu Xilinx ISE, a vestavěným logickým analyzátorem ChipScope. V simulátoru Test bench waveform (dále jen simulátor) jsou testovány jednotlivé části této práce. Ve vestavěném logickém analyzátoru ChipScope (dále jen logický analyzátor) je testována práce jako celek.

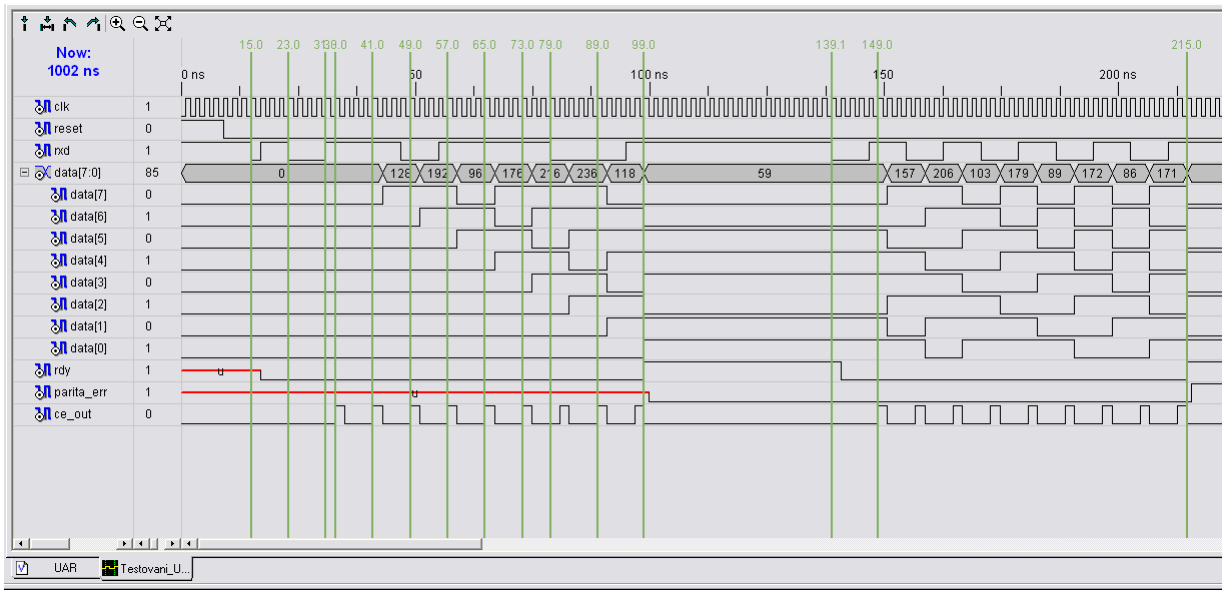
Simulátor nám dovoluje funkční analýzu projektu. Funkční simulace pracuje s ideálními vlastnostmi použitých logických prvků, s jejich nulovým zpožděním a také s nulovým zpožděním signálu na propojovacích signálových cestách mezi prvky. Funkční simulace je relativně rychlá. Zejména u složitějších návrhů s potenciální možností vzniku logických hazardů však nemusí podat pravdivé výsledky [6]. Proto je pro celkové testování využit nástroj pro diagnostiku reálného zapojení obvodu, logický analyzátor.

## 6.1. Simulace jednotlivých částí

### 6.1.1. Simulátor

Simulace projektu je rozdělena do jednotlivých částí tak, aby byly výsledky testování přehlednější. V simulátoru jsou vstupní data a signály do jednotlivých částí nastavena předem. Před spuštěním simulace je ještě nastavena hodnota doby simulace, délky trvání logické ‘1’ a ‘0’ a trvání náběžné, sestupné hrany signálu CLK.

#### 6.1.1.1. UAR

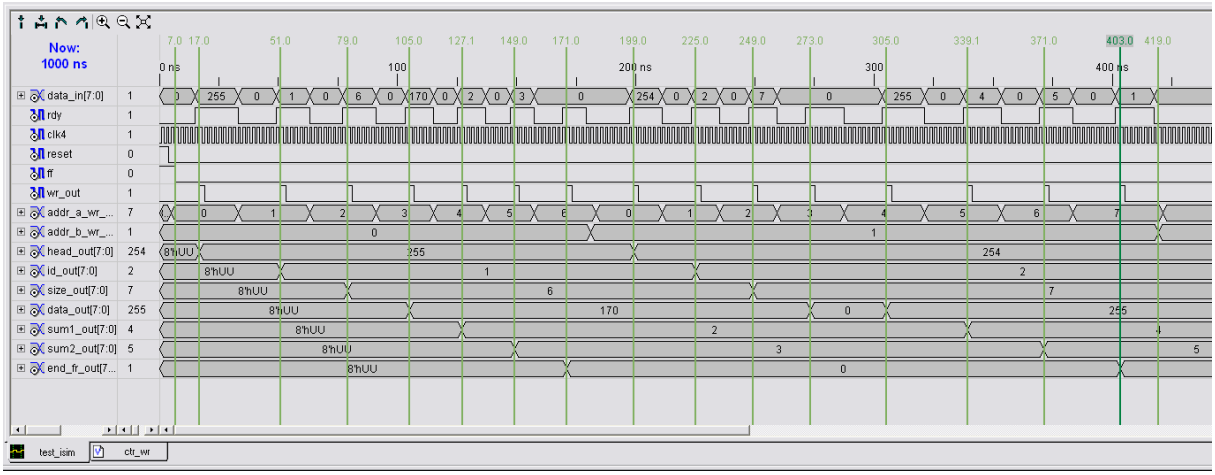



Obr. 5.1. Simulace bloku UAR – komponenta Test bench

Po resetu zařízení se čeká na logickou ‘0’ na lince RXD. V čase 15 ns byla simulována „falešná“ nula. Přijímač na ni reaguje, ale nepokládá ji za přijímací start bit, protože neobsahuje 4 stejné vzorky logické ‘0’. V čase 23 ns opět reaguje na další logickou ‘0’, ta už je pokládána za Start bit a začíná příjem dalších bitů (31 ns). V časech 33, 41, 49, 57, 65, 73, 8, 89 a 97 ns je vzorkován přijímaný bit a jeho hodnota je uložena do posuvného registru, plnícího se zleva (7 downto 0). V čase 99 ns, po navzorkování posledního přijímaného bitu, se nastaví signál RDY = ‘0’, signalizující připravenost dat na výstupní datové sběrnici DATA. V tento okamžik je spočítána kontrolní parita a v následující periodě hodin signalizuje správnost přijatých dat signál Parita_Err (parita sudá; správně

přijatý bajt = logická ‘0’ a naopak). Data zůstávají na datové sběrnici DATA tak dlouho, dokud se na lince neobjeví další logická ‘0’ Start_bit. Poté je signál RDY „shozen“ do nuly a příjem se opakuje (149 ns). V čase 215 ns, se zobrazuje na výstupu hodnota přijatého bajtu (viz řádky DATA vlevo obr. 5.1.).

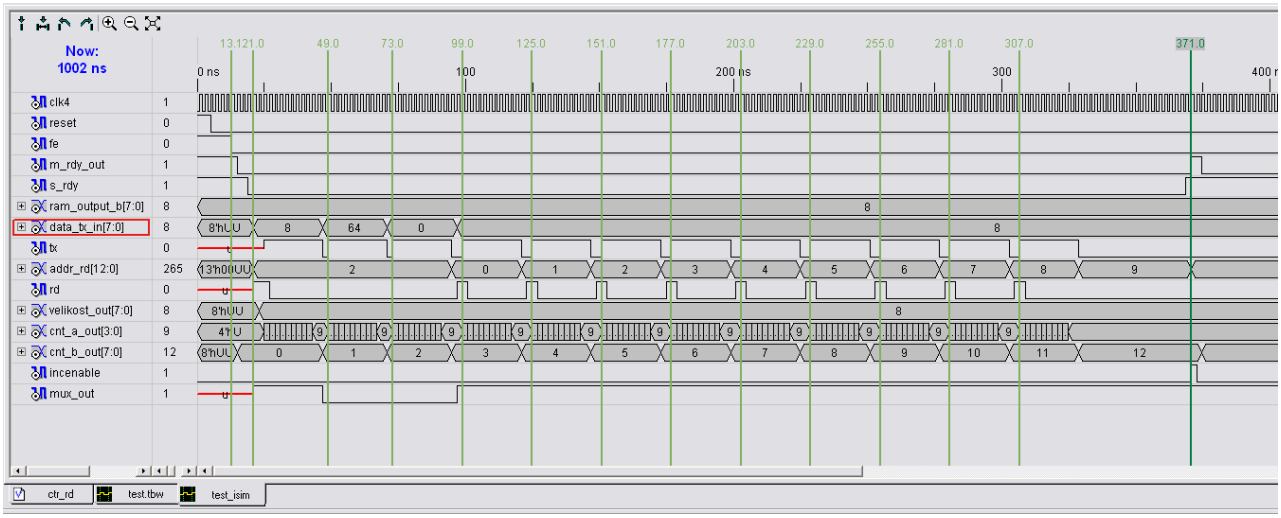
#### 6.1.1.2. CTR_WR




Obr. 5.2. Simulace bloku CTR_WR – komponenta Test bench

Simulace stavového automatu CTR_WR, pro zápis dat do FIFO, byla podobná jako u UAR. Na vstup se v různých časech nastavily hodnoty přijímaných bajtů a signál RDY=‘1’, signalizující připravenost dat na sběrnici DATA_IN (vstupu do FIFO). Po resetu a kontrole volného místa v paměti FF = ‘0’, signál FF byl nastaven také dopředu a signalizuje do času 7 ns stav plné FIFO. Při nástupné hraně RDY se data čtou a povoluje se zápis do paměti signálem WR_out v 17 ns a s nastavením adresy addr_a_wr. Jednotlivé části přijímaných bajtů jsou postupně, jak jsou načítány z UAR, zobrazovány v proměnných Head, ID, Size, Data (jen aktuální bajt), Sum1, Sum2 a End, tj. konce rámce. Tyto proměnné se mohou vyhodnocovat a podle nich se může řídit příjem i z jiných zařízení (viz Cerberos 2). V okně jsou zobrazeny dva přijaté rámce, druhý je zapisován v čase 199 ns po přičtení adresy počtu rámců v čase 180 ns.

### 6.1.1.3. CTR_RD

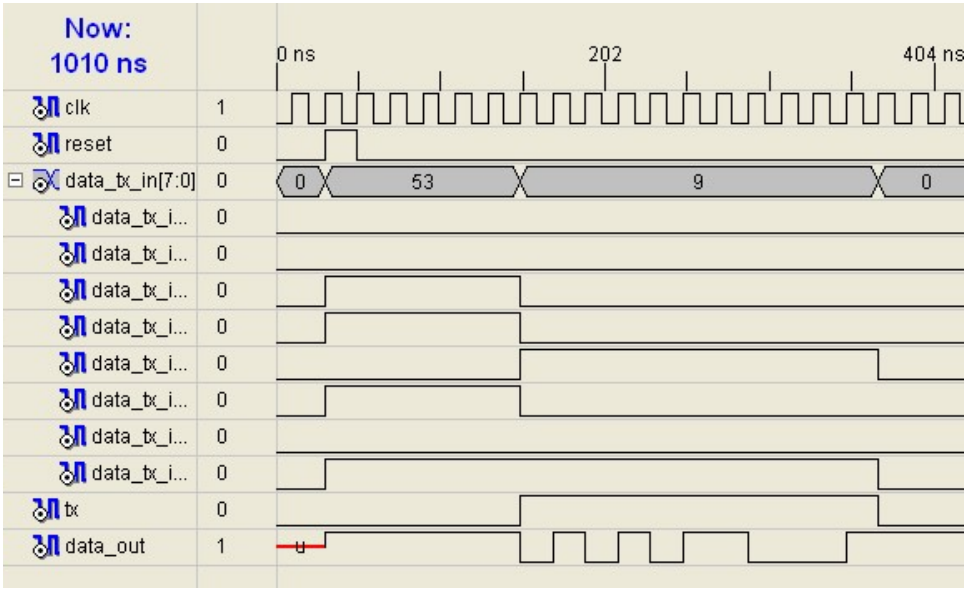



Obr. 5.3. Simulace bloku CTR_RD – komponenta Test bench

Simulace stavového automatu CTR_RD pro nastavení modulu ZigBee a čtení dat z FIFO byla podobná jako u předešlých testování. Předem se nastavily signály Reset, FE a S_RDY. Pro přehlednost je na obr. 5.3. simulace bloku část nastavení komunikace s modulem ZigBee a část vysílání dat. Tímto jsem si ověřil funkčnost jak nastavení, tak vysílání dat.

Po resetu celého zařízení v 8 ns se testují data ve FIFO, zda je v něm minimálně jeden rámec k odeslání. Pokud ano (FE = ‘0’, 13.1 ns), je nastaven signál M_RDY = ‘0’ (18 ns), tím dáváme najevo modulu ZigBee, že chceme vysílat data a čekáme na jeho odpověď připravení příjmutí dat, v podobě signálu S_RDY = ‘0’ (20 ns). Poté se přepne výběr multiplexoru (mux_out = ‘1’) na část čtení dat z FIFO a z adresy Addr_RD = 2, při povoleni čtení signálem RD = ‘1’, přečteme velikost vysílaných dat, kterou se nastaví celá část vysílání dat CTR_DR. Tato data (velikost) se také vyšlou jako první bajt do UAT. Poté se přepne multiplexor na nastavení ZigBee modulu (mux_out = ‘0’) a vyšlou se dva bajty CMD0 a CMD1 viz kapitola 2.4.1.2 SPI - Soubor příkazů Command do UAT. V následující 96 ns se přepne multiplexor zpět na čtení dat z paměti a od 98 ns se po 31ns postupně vysílají data z paměti z adresy Addr_RD než jsou všechna data odeslána. Po vyslání celého rámce dat se odečte počet z čítače rámců CNT_FIFO (371 ns) a čekáme na nastavení signálu S_RDY = ‘1’, tím signalizuje modul ZigBee příjmutí všech dat. Komunikace vysílání končí při nastavení M_RDY = ‘1’ a můžeme ji ihned opakovat.

#### 6.1.1.4. UAT



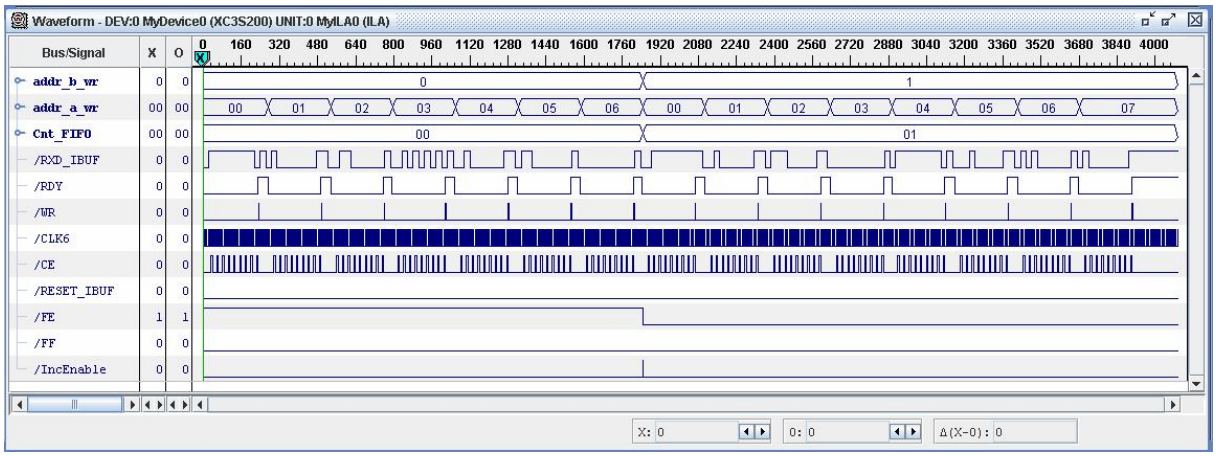

Obr. 5.4. Simulace bloku UAT – komponenta Test bench

Process vysílání dat je řízen časem signálu TX v logické ‘1’. Tato doba je dlouhá 11 CLK, vysílá se 11 bitů dat. Na obr. 5.4. jsou paralelní vstupní data (Data_TX_IN), která jsou serializována a vyslána na výstup Data_out bit po bitu.

### 6.1.2. Logický analyzátor ChipScope

V logickém analyzátoru se nastaví předem jednotlivé sledované výstupy, hodiny CLK, ke kterým se bude celý systém sledovat, včetně dalších hodnotově nižších CLK hodin (děličky hlavních hodin). Celý projekt se musí opět zkompilovat a nahrát do paměti vývojové desky Spartan 3. Poté je spuštěno sledování námi předem nastavených dat a signálů. Na vstup se přivede fyzicky přes RS232, pomocí např. Terminálu sériově vysílaný bajt dat (hexa znak), a ChipScope zobrazí po sledovanou dobu průběhy sledovaných signálů. V tomto logickém analyzátoru je simulována polovina práce, od vstupu dat po nahrání do paměti, a práce jako celek. Je tak učiněno z důvodu přehlednosti v počtu sledovaných signálů.

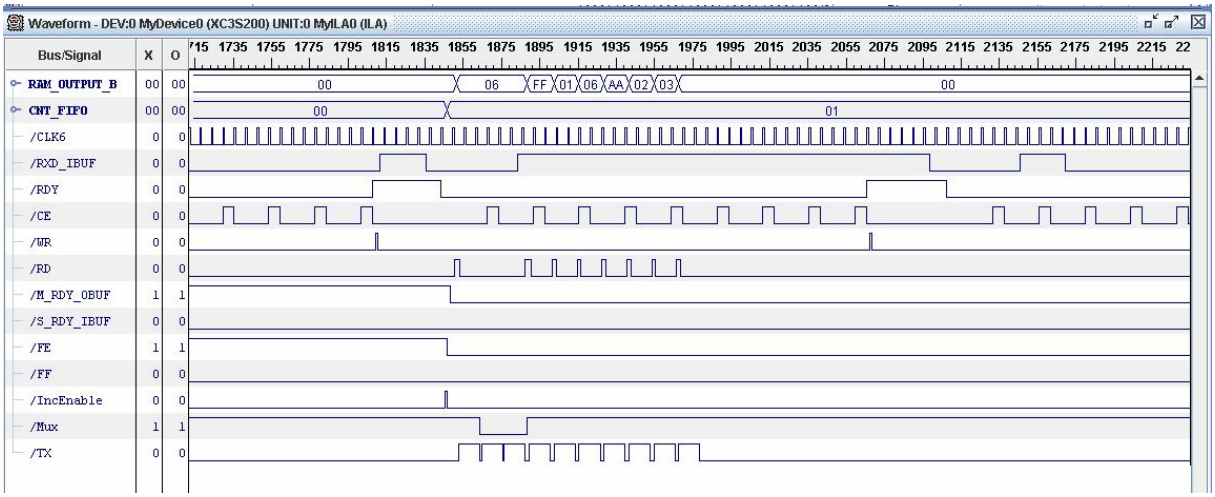
### 6.1.2.1. Diagnostika reálného zapojení 1/2 celkové práce




Obr. 5.5. Simulace 1/2 práce – Logický analyzátor

Na obr. 5.5. je vidět, jak se mění hodnoty jednotlivých adresových sběrnic a signálů. Addr_b_wr přičítání adresy příjmu (počtu rámců v paměti, horní části FIFO). Addr_a_wr počet jednotlivých přijímaných bajtů rámce (dolní část adresy do FIFO). RXD jsou data ze sériové linky (bity hexa znaků). Signál RDY =‘1’ povoluje čtení z bloku UAR (příjmu dat) a signál WR zápis jednotlivých bajtů dat (hexa znaků) do paměti FIFO. Jde velmi dobře vidět při této simulaci, jak se načítají jednotlivé bity (CE) a kdy se začne naplňovat čítač rámců v paměti (CNT_FIFO), tzn. v čase 1840 je přičten počet rámců a jsou signalizována data ve FIFO (FE = ‘0’).

### 6.1.2.2. Diagnostika reálného zapojení celkové práce




Obr. 5.6. Simulace celá práce – Logický analyzátor

V diagnostice celého návrhu, lze vidět vzorkování jednotlivých přijímaných bitů, tzn. že signál RXD trvá minimálně po dobu 4 CLK6. Další části pak pracují se 4 násobnou frekvencí, toto bylo

uzpůsobeno nutnosti vidět jednotlivé signály najednou. Proto byla maximální frekvence 4 MHz pro celý projekt (kromě UAR) upravena na frekvenci 115200 Hz. Zobrazovaný blok simulace byl posunut na čas, kdy už je vidět alespoň jeden rámec v paměti a nezávisle na příjmu je vysílán na výstup (data na sběrnici Ram_Output_B), signál přepínaní Mux čtení z paměti a nastavování modulu ZigBee a povolování vysílání pro proces UAT signálem TX = '1'. Pro malou rychlost vzorkování signálů nejde vidět vysílání jednotlivých bitů z UAT, ale tato část je vidět v předešlé simulaci jednotlivého bloku (5.1.1.4).

# 7. Rozšíření pro Cerberos 2

Tato diplomová práce bude v budoucnu rozšířena o další součásti (přenos různých dat) pro „Cerberos 2". Návrh, jak toto zařízení rozšířit, je nastíněno v následující kapitole (7.1.).

## 7.1. Příjem různých dat

Nastavení příjmu a vysílání stavových automatů, pro příjem jiných dat než která jsou zpracovávána v této práci, může být řešen novým, nadřazeným stavovým automatem nebo se ke každému stávajícím (CTR_WR a CTR_RD) přidá nový stav kontroly Identifikátoru a podle jeho hodnoty se rozhodne o nastavení jejich parametrů. Hlavním parametrem je velikost přijímaných dat, protože jejich délka je různá, musí se zajistit jejich správná doba pro příjem/vysílání. Jedná se o příjem/vysílání dat protokolu pro Cerberos 2 viz příloha.

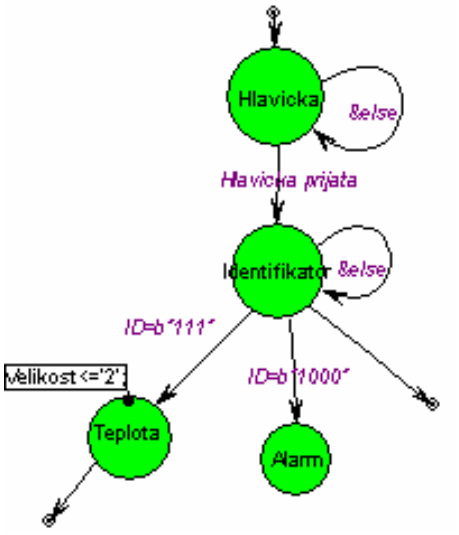
### 7.1.1. Nadřazený automat

Využitím nadřazeného automatu ke stávajícím (CTR_WR a CTR_RD) se nevyhneme ztrátě dat při příjmu. Jedná se o informace o hlavičce, identifikátoru a patičce s kterými by nadřazený automat pracoval. Změní se tedy koncept této práce, nebude využívána jako „roura" pro přenos dat. Pokud by jsme chtěli rámec zachovat, museli bychom jej na výstupu rozšířit o „ztracená" data ze statické adresy. Takovýto automat by po detekci hlavičky a identifikátoru, nastavil délku přijímaného/ vysílaného rámce a nastavil signálem CTR_WR_EN jeho uložení do paměti (kapitola 5.4.) a signálem

CTR_RD_EN odeslání (kapitola 5.6.) do ZigBee modulu a čekal na zpětnou odezvu o ukončení uložení/vyslání. Musí se zajistit nezávislost uložení a odeslaní, ale nadřazený automat musí vědět která data přijímá/vysílá, aby nedošlo k jejich přepsání/ špatnému přečtení z paměti FIFO.

### 7.1.2. Rozšíření stávajících automatů

Rozšíření stávajících automatů (CTR_WR a CTR_RD) o nový stav kontroly Identifikátoru je na obr.7.1. Podle jeho hodnoty se rozhodne o nastavení jejich parametrů. Hlavním parametrem je velikost přijímaných dat, která se nastaví po vyhodnocení identifikátoru. U vysílání dat je nutné načtení identifikátoru z paměti, zjištění velikosti dat a výpočet délky celého rámce pro vyslání.




Obr.7.1. Příjem různých dat modulem FPGA

# 8. Závěr

Úkol zadaný na začátku diplomové práce byl splněn. Výsledkem jsou zdokumentované technologie ZigBee, FPGA, síťový prvek transceiver CC2480 a vytvoření funkčního bloku rozhraní ZigBee pro modulární telemetrický systém „Cerberos“ implementovaný v FPGA obvodu Spartan 3 s využitím vývojové desky (Digilent Starter kit) a jejími periferiemi (RS232, SRAM, přepínačů a diod).

V první části jsou popsány již zmíněné technologie. Na poli využitelných bezdrátových komunikačních technologií vyplňuje mezeru v podobě velké skupiny aplikací, pro které nejsou Bluetooth, ani WiFi, příp. Irda ideálním řešením, bezdrátový standard ZigBee a, jeden z nejvýznamnějších směrů vývoje integrovaných obvodů s velmi velkou hustotou integrace, programovatelné hradlové pole FPGA. Vzhledem k tomu, že I/O u FPGA je v zásadě standardní binární logika, bylo potřeba vyřešit nejnižší vrstvy komunikačního protokolu „off the Chip“, vrstvy MAC a PHY komunikačního protokolu ZigBee. Z tohoto důvodu byl v této práci použit transceiver CC2480.

V hlavní části je vytvořen protokol přenosu přijímaných/vysílaných dat, probíhající jedním směrem, ze sériové linky SCI přes FPGA modul do trensceiveru CC2480 ZigBee modulu na jednom plošném spoji. Z tohoto návrhu jsem vycházel při tvorbě modulu FPGA a jeho částí. V první fázi jsem vytvořil návrh celého zařízení a postupně popisoval jednotlivé bloky v jazyku typu VHDL tak, jak je přijímaný rámec zpracováván, ukládán do paměti FIFO a následně odesílán do ZigBee modulu. Tímto přístupem se původní celkový návrh doplňoval o další, většinou kontrolní, bloky a signály. Důležitou částí bylo správné časování jednotlivých bloků a v konečném důsledku i synchronizování jednotlivých logických členů k jediným a základním hodinám o frekvenci 50MHz. Takto se minimalizovalo zpoždění jednotlivých obvodů ze vstupu na výstup a předešlo se možným hazardům.

Každý vytvořený blok byl přínosem ve smyslu nastudování jeho reálných vlastností a navržení do logického obvodu FPGA, popsaného v jazyce VHDL. Funkčnost vytvořených obvodů byla ověřena v simulátoru Test bench waveform, komponenty programu Xilinx ISE, a vestavěným logickým analyzátorem ChipScope. Všechny části (i jako celek) podle těchto testovacích programů pracují správně, bez chyb a podle očekávaní. Jedinou nevyřešenou součástí bylo samotné vyzkoušení se zařízením ZigBee, připojeným k vývojové desce Spartanu 3, neboť nebylo možné ověřit funkčnost tohoto zařízení. Nepodařilo se nastavit komunikaci s opačným přijímajícím zařízením.

Tato diplomová práce bude rozšířena o další specifika použití pro rozhraní ZigBee modulárního telemetrického systému „Cerberos 2“, který bude dokončen začátkem roku 2010. Návrhy na rozšíření této práce jsou nastíněny v kapitole 7 ( rozšíření pro Cerberos 2).

# Literatura

[1] Pinker, Jiří, Poupa, Martin. Číslicové systémy a jazyk VHDL, Ben – technick8 literatura, 2006. ISBN 80-7300-198-5

[2] Bradáč, Zdeněk. Bezdrátový komunikační standard ZigBee [online]. [2005] [cit. 2009-01-12]. Dostupný z WWW: <http://www.automatizace.cz/article.php?a=638>.

[3] Daněk, Martin. Programovatelná hradlová pole – FPGA [online]. [2006] [cit. 2009-03-24].
Dostupný z WWW: <http: //www.odbornecasopisy.cz/index.php?id_document=30930>.

[4] FPGA : Popis produktu [online]. [2007] [cit. 2009-03-24]. Dostupný z WWW: <http://www.xilinx.com/support/documentation/spartan-3.htm>.

[5] Spartan-3 Starter Kith : Popis produktu [online]. [2009] [cit. 2009-04-24]. Dostupný z WWW: <http://www.xilinx.com/support/documentation/spar3-sk.htm>.

[6] Kašík,Vladimír. Prostředky pro navrhování logických systémů pro obvody FPGA [online]. [2006] [cit. 2009-03-12]. Dostupný z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=31260>.

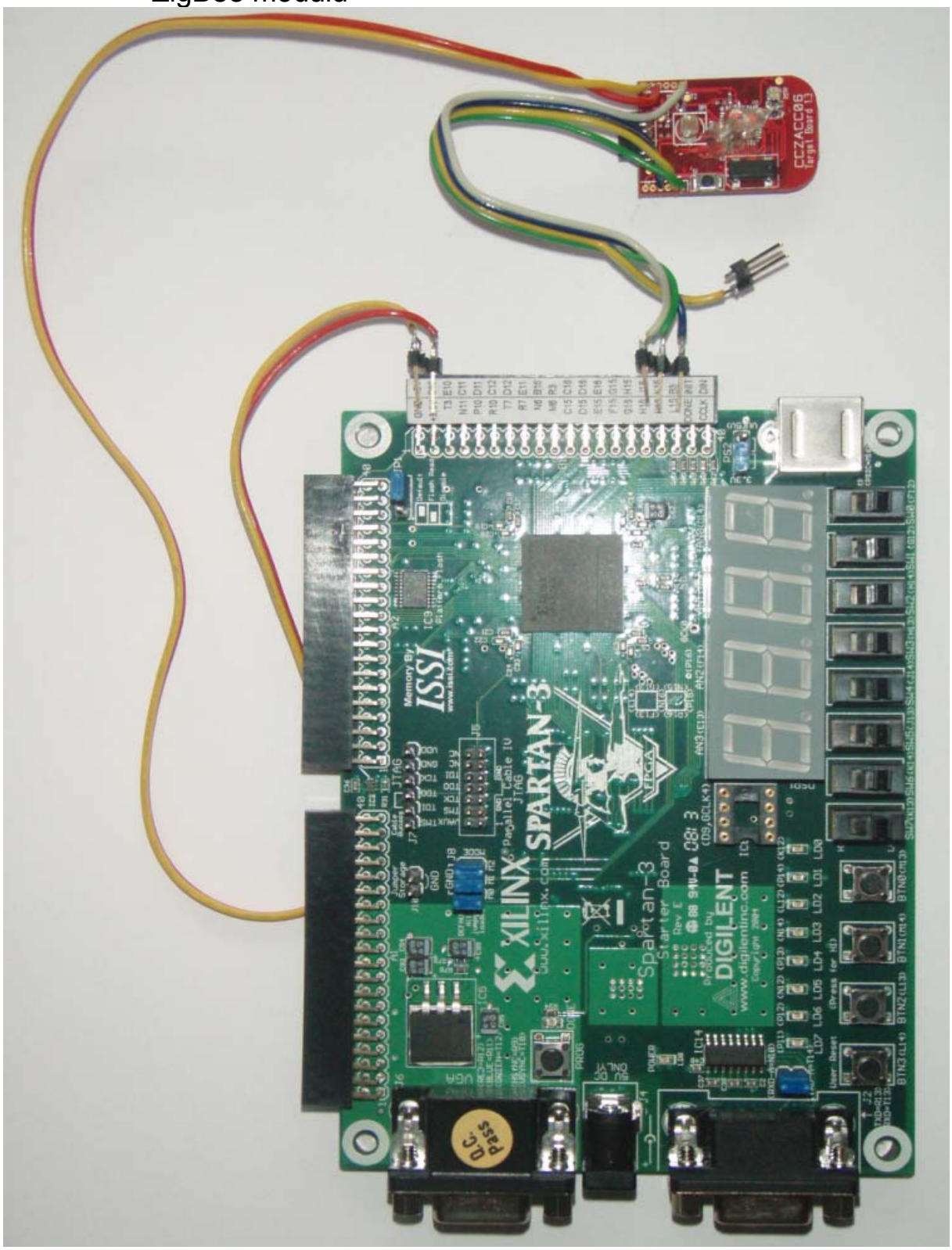
[7] Texas Instruments. Transceiver CC2480 Data Sheet, [cit. 2009-03-24]. Dostupný z WWW: <http://focus.ti.com/docs/prod/folders/print/cc2480a1.html>.

[8] Parknell,K.,Mehta,N.: Programmble Logic Design Quick Start Handbook, Xilinc Inc.,2003

[9] Ashenden, P.: The Designer’s Guide to VHDL. Morgan Kaufmann Publischers, 1998.ISBN 1-55860-270-4

# Seznam příloh

Příloha I – Fotografie vývojové desky osazené obvodem Spartan 3 a ZigBee modulu



## Příloha II –Výpis kódu modulu do FPGA

```
entity diplomka is
	generic(
		MAX_CNT_109	: INTEGER := 109-1;		-- delitel 50MHz
		MAX_CNT_6: INTEGER := 6-1;			-- delitel 4x115200Hz

		CNT_13: integer	:= 12;				-- delitel 13
		max	: integer	:= 16
		);
	Port (
		CLK		: in STD_LOGIC;				-- hodiny
		RESET		: in STD_LOGIC;				-- reset
		Rychlost	: in std_logic_vector(2 downto 0);	-- vyber rychlosti prijmu
		RXD		: in STD_LOGIC;				-- vstupni data
		PARITA_ERR: out std_logic;
		S_RDY : in STD_LOGIC;			-- příznak připravenosti ZigBee
		M_RDY : out STD_LOGIC;			-- nastavení vysílání do ZigBee
		Data_out : out STD_LOGIC;		-- výstupní data
		);
end diplomka;

architecture diplomka of diplomka is

		signal DATA  : std_logic_vector(7 downto 0);	-- vystupni data z uar
		signal DATA_IN: std_logic_vector ( 7 downto 0 ); -- vstupni data
		signal RDY	: std_logic;		-- DATA_IN jsou pripravena k zapisu


-- signaly delicky clk4M
		signal CNT	: std_logic_vector (3 downto 0);
		signal CLK4	: std_logic;


-- signály FIFO
		type memA is array (0 to (max * 256)-1) of std_logic_vector ( 7 downto 0);
		signal fifo_mem : memA;

		signal WR		: STD_LOGIC;
		signal RD		: STD_LOGIC;

		signal ADDR_WR: STD_LOGIC_VECTOR (11 downto 0); -- adresa zapisu
		signal ADDR_RD: STD_LOGIC_VECTOR (11 downto 0);-- adresa cteni
		signal RAM_OUTPUT_A : STD_LOGIC_VECTOR (7 downto 0);-- vystup A
dat z pameti
```

```
		signal RAM_OUTPUT_B : STD_LOGIC_VECTOR (7 downto 0);-- vystup B
dat z pameti


-- čítač dat ve FIFO, nastavení FF a FE
		signal FF : std_logic;				-- ff  = fifo full  = plny
		signal FE : std_logic;			-- ff  = fifo empty = prazdny
		signal IncEnable : std_logic;		-- signál pro přičtení počtu prvků v paměti
		signal DecEnable : std_logic;		-- signál pro odečtení počtu prvků v
paměti
		signal Cnt_FIFO:	std_logic_vector (4 downto 0);


-- signal FSM CTRL_WR

		type Sreg0_type is (RST, S1, S2, S3, S4, S5, S6, S7, S8, Wait_data_IN,
Wait_End, Wait_hd, Wait_ID, Wait_Size, Wait_sum, wr_DATA_IN, Wr_End, wr_hd,
wr_ID, wr_Size, wr_Sum1, wr_Sum2);
		signal Sreg0: Sreg0_type;

		signal addr_a_wr	: STD_LOGIC_VECTOR (7 downto 0);
		signal addr_b_wr	: STD_LOGIC_VECTOR (3 downto 0);
		signal SIZE: STD_LOGIC_VECTOR (7 downto 0);		-- velikost


-- signal FSM CTRL_RD

		type CTR_RD_type is (CNT_DATA, S1_M_RD, S1_RD, S2_RD, S_CMD0,
S_CMD1, S_DATA, S_TX, S_TX0, S_Velikost, Set, SIZE_addr, Wait_DATA_OUT,
Wait_S_RDY, wait_S_rdy_end);
		signal CTR_RD: CTR_RD_type;

		signal addr_a_rd	: STD_LOGIC_VECTOR (7 downto 0);
		signal addr_b_rd	: STD_LOGIC_VECTOR (3 downto 0);

		signal Velikost: STD_LOGIC_VECTOR (7 downto 0);	-- velikost dat

		constant CMD0: STD_LOGIC_VECTOR (7 downto 0):=b"01000000"; --
nastavení přenosu mezi FPGA a ZigBee, hodnota 01000000 známená přenos AREQ
		constant CMD1: STD_LOGIC_VECTOR (7 downto 0):=b"00000000"; --
nastavení přenosu mezi FPGA a ZigBee, tento bajt musí být roven nule
		signal CMD_TX_IN: STD_LOGIC_VECTOR (7 downto 0);
		signal CNT_A: STD_LOGIC_VECTOR (3 downto 0);	-- čítač 11 implusů
pro vysílač
		signal CNT_B: STD_LOGIC_VECTOR (7 downto 0);	-- čítač vyslaných
dat
```

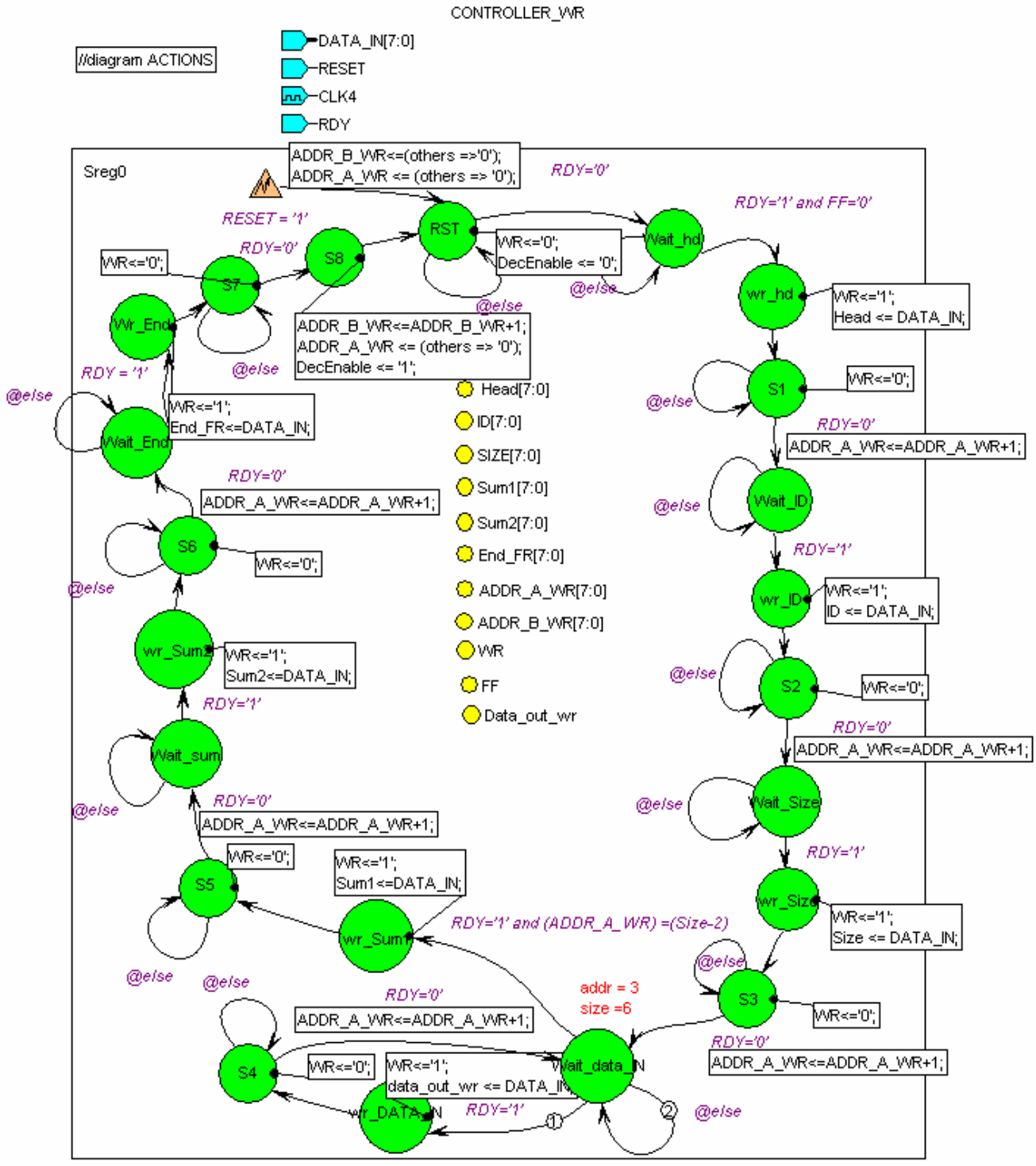
```
        signal CNT_CAS: STD_LOGIC_VECTOR (25 downto 0);-- čítač času čekání
na ZigBee
        signal Mux: STD_LOGIC;                  -- výběr dat FIFO x CMD_TX_IN
        signal Vel : std_logic;                 -- muze se smazat


-- signály vysílače UAT
        signal DATA_TX_IN: STD_LOGIC_vector(7 downto 0);  -- vstupní data
vysílače
        signal TX : STD_LOGI                    -- povolení vysílání UAT
        signal data_t: std_logic_vector( 10 downto 0);-- data vysílače
        signal par_t : std_logic;               -- výpočet parity
        constant start_bit      : std_logic:= '0';      -- start bit
        constant stop_bit       : std_logic:='1'        -- stop bit


-- signalz UAR
        signal CLK109 : std_logic;    -- delicka CLK109 = ( CNT / 109  ( 50MHz/109
= 4*115200 Hz )   )
        signal CNT_del_109  : STD_LOGIC_VECTOR (6 downto 0);      -- sbernice
MAX_CNT pro delicku 109
        signal CLK6: std_logic;         -- delicka CLK6C = NT4 / 6  ( = 4*19200 Hz )
        signal CNT_del_6      : STD_LOGIC_VECTOR (2 downto 0);      -- sbernice
MAX_CNT pro delicku 6
         signal Rychlost_OUT: std_logic; -- čítač rychlostí 9600, 4800, 2400, 1200,
600, 300
        signal Q: STD_LOGIC_VECTOR (5 downto 0); -- sbernice pro citac rychlosti
        signal CLK4_r: std_logic; -- hodiny pro UAR ( stejne jako Rychlost_OUT ) //
4 vzorky
        type FSM_type is (IDLE, Stav_1, Stav_2, Stav_3);          -- stavovy automat
        signal FSM: FSM_type;                                     -- stavovy automat

         signal CNT_r: STD_LOGIC_VECTOR (1 downto 0);       -- citac stavoveho
automatu
        signal Sample_CNT: STD_LOGIC_VECTOR (5 downto 0); -- citac stavoveho
    automatu
        signal PARITY : STD_LOGIC;                        -- čas povoleni parity
        signal CE : STD_LOGIC;                            -- čas povoleni vzorku bitu
        signal reg : std_logic_vector (8 downto 0);       -- posuvny registr
        signal Data_reg : std_logic_vector (8 downto 0);  -- vystupni data posuvneho
        registru pro vypocet parity
        signal P_ERR : std_logic;                         -- vypočet parity
        signal Par : std_logic;
begin
```

# Příloha III – Stavový automat příjmu dat (CTR_WR)



# Příloha IV – Stavový automat vysílání dat (CTR_RD)